User Manual

3-Heights™
PDF Security Shell

PDF/A compliant

Version 4.5

PDF-TOOLS.COM
Premium PDF Technology

Contents

1 Introduction . . . 1
1.1 Description . . . 1
1.2 Functions . . . 2
1.3 Operating Systems . . . 3
1.4 Digital Signatures . . . 3
What is an Electronic Signature? . . . 4
How to Create Electronic Signatures . . . 5
2 Installation And Deployment . . . 7
2.1 Windows . . . 7
2.2 Unix . . . 7
2.3 Note about the Evaluation Version . . . 8
3 License Management . . . 8
3.1 Graphical License Manager Tool . . . 8
3.2 Command Line License Manager Tool . . . 9
3.3 License Key Storage . . . 9
4 Getting Started . . . 10
4.1 Basics . . . 10
4.2 General Settings . . . 10
-lk: Set License Key . . . 10
4.3 Specify the Folder of the Output File . . . 10
4.4 Encryption . . . 10
4.5 Cryptographic Provider . . . 13
PKCS#11 Provider . . . 13
Microsoft CryptoAPI Provider . . . 15
3-Heights™ Signature Creation and Validation Service . . . 18
SwissSign Personal Signing Service . . . 20
SwissSign SuisseID Signing Service . . . 21
QuoVadis sealsign . . . 22
Swisscom All-in Signing Service . . . 23
4.6 How to Create Digital Signatures . . . 24
How to Create a PAdES LTV Signature . . . 24
How to Create a Time-stamp Signature . . . 24
How to Create a Visual Appearance of a Signature . . . 25
Miscellaneous . . . 25
4.7 How to Validate Digital Signatures . . . 27
4.8 Fonts . . . 29
5 Reference Manual . . . 30
5.1 Encryption . . . 30
-fe: Force Encryption . . . 30
-fr: Set String Crypt Filter . . . 30
-fm: Set Stream Crypt Filter . . . 30
-id: Add entries to the info object (metadata) . . . 31
-k: Set the Length of the Encryption Key . . . 31
-o: Set the Owner Password . . . 31
-p: Set the Permission Flags . . . 31
-pw: Read an Encrypted PDF File . . . 32
-u: Set User Password . . . 32
5.2 Digital Signatures . . . 32
-ap: Signature Page Number . . . 32
-ar: Signature Annotation Rectangle . . . 32
-cn: Certificate Name (Subject) . . . 33
-cr: Signature Reason . . . 33
-cci: Signer contact info . . . 33
-ci: Certificate Issuer . . . 33
-cno: Certificate Serial Number . . . 33
-cfp: Certificate Fingerprint . . . 33
-co: Do not Embed Revocation Information . . . 34
-cp: Cryptographic Provider . . . 34

-cps: Cryptographic session property (string) . . . 34
-cpf: Cryptographic session property (file) . . . 34
-csl: Certificate Store Location . . . 35
-csn: Certificate Store Name . . . 35
-tsu: Time-stamp URL . . . 35
-tsc: Time-stamp Credentials . . . 35
-wpu: Web Proxy Server URL . . . 35
-wpc: Web Proxy Server Credentials . . . 35
-nc: Disable cache for CRL and OCSP . . . 35
-af1: Signature Font Name 1 . . . 35
-af2: Signature Font Name 2 . . . 36
-at1: Signature Text 1 . . . 36
-at2: Signature Text 2 . . . 36
-abg: Signature Background Image . . . 36
-al: Signature Line Width . . . 36
-acf: Signature Fill Color . . . 36
-acs: Signature Stroke Color . . . 36
-dts: Create a Time-stamp Signature . . . 37
-mdp: Create a DocMDP Signature . . . 37
-dap: Document Access Permissions for DocMDP Signature . . . 37
-fs: Force Signature . . . 37
-p2f: Replace placeholder image with signature field . . . 37
-vs: Verify signature . . . 37
5.3 General Switches . . . 37
-lk: Set License Key . . . 37
-s: Add Stamps . . . 37
-v: Verbose Mode . . . 37
5.4 Return Codes . . . 38
6 Stamping . . . 39
7 Licensing and Copyright . . . 45
8 Contact . . . 45

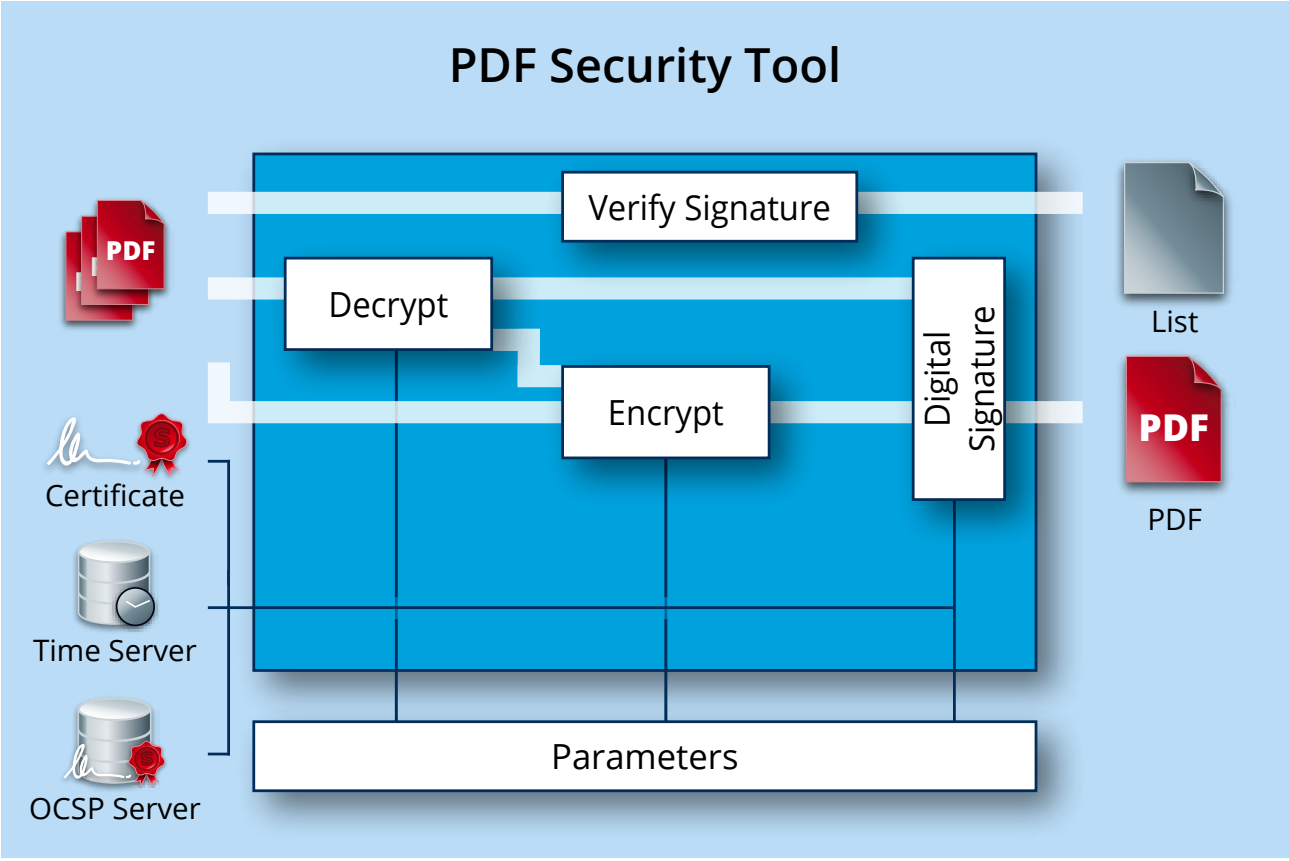
1 Introduction

1.1 Description

The 3-Heights™ PDF Security Shell enables the application of digital signatures to PDF documents and their subsequent protection through setting passwords and user authorizations.

Both standard signatures and qualified signatures that use signature cards (“smart cards”, “USB tokens”, “HSM”) can be used.

PDF documents used in professional circumstances contain important information that needs to be protected against misuse and unintentional alteration. This is achieved by protecting PDF documents through encryption and user authorization rights.



When exchanging electronic documents the ability to ascertain that a document is authentic and has not been manipulated on its way from sender to recipient is of particular importance. This is only achievable through the use of electronic signatures.

1.2 Functions

The 3-Heights™ PDF Security Shell enables users to encrypt and - if the passwords are known - decrypt PDF documents. The tool can set and cancel all known PDF user authorizations. It can, for instance, set an owner password so that only authorized users can edit and change the document. A user password ensures that only authorized users have access to the document's content. The tool's signature module allows the user to apply, read and verify both classic digital signatures and MDP (modification detection and prevention) signatures. The visibility and visual appearance of digital signatures can be adapted to suit requirements. The tool also supports customized signature handlers and types.

Features

- Apply simple, advanced and qualified electronic signatures
- Apply PAdES LTV (Long Term Validation) signatures
- Cache OCSP, CRL and TSP responses for mass signing
- Apply modification detection & prevention (MDP) signatures
- Apply document time-stamp signatures
- Encrypt and decrypt PDF documents
- Set user authorizations, including:
  - Print document
  - Modify document content
  - Extract or copy content
  - Add comments
  - Fill in form fields
  - Content extraction for accessibility
  - Assemble documents
  - Print in high resolution
- Set crypt and stream filters
- Set encryption strength
- Set owner and user password

Formats

Input Formats:

- PDF 1.x (e.g. PDF 1.4, PDF 1.5)
- PDF/A-1, PDF/A-2, PDF/A-3

Target Formats:

- PDF 1.x (e.g. PDF 1.4, PDF 1.5)
- PDF/A-1, PDF/A-2, PDF/A-3

Compliance

Standards: ISO 32000 (PDF 1.7), ISO 19005-1 (PDF/A-1), ISO 19005-2 (PDF/A-2) , ISO 19005-3 (PDF/A-3), PAdES Part 2 and Part 4 (Long Term Validation, LTV)

1.3 Operating Systems

- Windows XP, Vista, 7, 8, 8.1 - 32 and 64 bit
- Windows Server 2003, 2008, 2008 R2, 2012, 2012 R2 - 32 and 64 bit
- HP-UX 11 and later PA-RISC2.0 32 bit or HP-UX 11i and later ia64 (Itanium) 64 bit
- IBM AIX 5.1 and later (64 bit)
- Linux (32 and 64 bit)
- Mac OS X 10.4 and later (32 and 64 bit)
- Sun Solaris 2.8 and later, SPARC and Intel
- FreeBSD 4.7 and later 32 bit or FreeBSD 9.3 and later 64 bit (on request)

1.4 Digital Signatures

Overview

Digital signature is a large and slightly complex topic. This manual gives an introduction to digital signatures and describes how the 3-Heights™ PDF Security Shell is used to apply them. It does however not describe all the technical details.

Terminology

Digital Signature is a cryptographic technique of calculating a number (a digital signature) for a message. Creating a digital signature requires a private key from a certificate. Validating a digital signature and its authorship requires a public key. Digital Signature is a technical term.

Electronic Signature is a set of electronic data that is merged or linked to other electronic data in order to authenticate it. Electronic Signatures can be created by means of a digital signature or other techniques. Electronic Signature is a legal term.

Table: Abbreviations	
CA	Certification Authority
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
CSP	Cryptographic Service Provider
HSM	Hardware Security Module
OCSP	Online Certificate Status Protocol
PKCS	Public Key Cryptography Standards
QES	Qualified Electronic Signature
TSA	Time-stamp Authority
TSP	Time-stamp Protocol

Why Digitally Signing?

The idea of applying a digital signature in PDF is very similar to a handwritten signature: A person reads a document and signs it with their name. In addition to the name, the signature can contain further optional information, such as the date and location. A valid electronic signature is a section of data that can be used to:

- Ensure the integrity of the document
- Authenticate the signer of the document
- Prove existence of file prior to date (time-stamp)

Digitally signing a document requires a certificate and its private key. How to access and use a certificate is described in the chapter Cryptographic Provider.

In a PDF document, a digital signature consists of two parts:

- *A PDF related part*: This part consists of the PDF objects required to embed the signature into the PDF document. This part depends on the signature type (Document Signature, MDP Signature, see table below). Information such as name of the signer, reason, date, location is stored here. The signature may optionally have a visual appearance on a page of the PDF document, which can contain text, graphics and images. This part of the signature is entirely created by the 3-Heights™ PDF Security Shell.
- *A cryptographic part*: A digital signature is based on a cryptographic checksum (hash value) calculated from the content of the document that is being signed. If the document is modified at a later time, the computed hash value is no longer correct and the signature becomes invalid, i.e. the validation will fail and will report that the document has been modified since the signature was applied. Only the owner of the certificate and its private key is able to sign the document. However, anybody can verify the signature with the public key contained in the certificate. This part of the signature requires a cryptographic provider for some cryptographic data and algorithms.

The 3-Heights™ PDF Security Shell supports the following types of digital signatures:

- *Document Signature*: Check the integrity of the signed part of the document and authenticate the signer's identity. One or more signatures can be applied. A signed document can be modified and saved by incremental update. The state of the document can be re-created as it existed at the time of signing.
- *MDP (Modification detection and prevention) Signature*: Enable detection of disallowed changes specified by the author. A document can contain only one MDP signature; it must be the first in the document. Other document signatures may be present.
- *Document Time-stamp Signature*: Establish the exact content of the file at the time indicated by the Time-stamp. One or more document Time-stamp signatures can be applied. A signed document can be modified and saved by incremental update.

1.4.1 What is an Electronic Signature?

There are different types of electronic signatures, which normally are defined by national laws, and therefore are different for different countries. The type of electronic signatures required in a certain process is usually defined by national laws. Quite advanced in this manner are German-speaking countries where such laws and an established terminology exist. The English terminology is basically a translation from German. Three types of electronic signatures are distinguished:

- Simple Electronic Signature "Einfache Elektronische Signatur"
- Advanced Electronic Signature "Fortgeschrittene Elektronische Signatur"
- Qualified Electronic Signature (QES) "Qualifizierte Elektronische Signatur"

All applied digital signatures are PDF/A and PAdES compliant.

Simple Electronic Signature

A simple electronic signature requires any certificate that can be used for digital signing. The easiest way to retrieve a certificate, which meets that requirement, is to create a so called self-signed certificate. Self-signed means it is signed by its owner, therefore the issuer of the certificate and the approver of the legitimacy of a document signed by this certificate is the same person.

Example: Anyone could create a self-signed certificate issued by "Peter Pan" and issued to "Peter Pan". Using this certificate one is able to sign in the name of "Peter Pan".

If a PDF document is signed with a simple electronic signature and the document is changed after the signature has been applied, the signature becomes invalid. However, the person who applied the changes, could at the same time (maliciously) also remove the existing simple electronic signature and - after the changes - apply a new, equally looking Simple Electronic Signature and falsify its date. As we can see, a simple electronic signature is neither strong enough to ensure the integrity of the document nor to authenticate the signer.

This drawback can be overcome using an Advanced or Qualified Electronic Signature.

Advanced Electronic Signature

Requirements for advanced certificates and signatures vary depending on the country where they are issued and used.

An advanced electronic signature is based on an advanced certificate that is issued by a recognized certificate authority (CA) in this country, such VeriSign, SwissSign, QuoVadis. In order to receive an advanced certificate, its owner must prove his identity, e.g. by physically visiting the CA and presenting his passport. The owner can be an individual, a legal person or another entity.

An advanced certificate contains the name of the owner, the name of the CA, its period of validity and other information.

The private key of the certificate is protected by a PIN, which is only known to its owner.

This brings the following advantages over a simple electronic signature:

- The signature authenticates the signer.
- The signature ensures the integrity of the signed content.

Qualified Electronic Signature

Requirements for qualified certificates and signatures vary depending on the country where they are issued and used.

A Qualified Electronic Signature is similar to an advanced electronic signature, but has higher requirements. The main differences are:

- It is based on a qualified certificate, which is provided as a hardware token (USB stick, smart card).
- For every signature it is required to enter the PIN code manually. This means that only one signature can be applied at a time.
- Certificate revocation information (OCSP/CRL) can be acquired from an online service. The response (valid, revoked, etc.) must be embedded in the signature.
- A time-stamp (TSP) that is acquired from a trusted time server (TSA) may be required.

This brings the following advantages over an advanced electronic signature:

- The signature ensures the certificate was valid at the time when the document was signed (due to the embedding of the OCSP/CRL response).
- The signature ensures the integrity of the time of signing (due to the embedding of the time-stamp).
- Legal processes that require a QES are supported.

Note that a Time-stamp can be added to any type of signature. OCSP/CRL responses are also available for some advanced certificates.

1.4.2 How to Create Electronic Signatures

Preparation Steps

1. Identify whether an advanced or a qualified signature is required. For most automated processes an advanced signature is sufficient.
2. Acquire a corresponding certificate from a CA. Note that some CA offer USB sticks or smart cards that contain both, an advanced and a qualified certificate.
3. Setup and configure the certificate's Cryptographic Provider
   - In case the certificate resides on hardware such as an USB token or a Smart Card, the required middleware (driver) needs to be installed.

- In case the certificate is a soft certificate, it must be imported into the certificate store of a cryptographic provider.

4. Optional: Acquire access to a trusted time server (TSA) (e.g. from the CA of your signing certificate).
5. Apply the signature by providing the following information:
   - Values for the selection of the signing certificate (e.g. the name of the certificate)
   - The Cryptographic Provider where the certificate is located
   - Optional: Time-stamp service URL (e.g. "http://server.mydomain.com:80/tsa")
   - Optional: Time-stamp service credentials (e.g. username:password)
   - Optional: Embed revocation information (default: true)
   - Optional: Visual appearance of the signature on a page of the document (e.g. an image).

Example: Steps to Add an Electronic Signature

The 3-Heights™ PDF Security Shell applies PDF/A compliant signatures. This means if a PDF/A document is digitally signed, it remains PDF/A compliant.

In order to add an electronic signature with the 3-Heights™ PDF Security Shell the following steps need to be done:

1. Provide the certificate name (Subject)
2. Apply settings for the signature, such as the reason text, or the visual appearance (color, position, etc).
3. Process the PDF document by a user which has access to the selected certificate and thereby add the signature

The certificate name is provided with the switch -cn, the reason with the switch -cr and the provider (including the PIN to access the certificate's private key) with the switch -cp. A sample command looks like this:

```
pdfsecure
-cn "Philip Renggli"
-cp "cvp11.dll;0;secret-pin"
-cr "I reviewed the document"
-tsu "http://server.mydomain.com:80/tsa"
input.pdf output.pdf
```

Note: On Non-Windows platforms additional settings are required to be set, such as fonts.

The visual appearance of the digital signature on a page of the resulting output-document looks as shown below:

2 Installation And Deployment

2.1 Windows

The retail version of the 3-Heights™ PDF Security Shell comes as a ZIP archive containing various files including runtime binary executable code, files required for the developer, documentation and license terms.

1. Download the ZIP archive of the product from your download account at http://www.pdf-tools.com.
2. Unzip the file using a tool like WinZip available from WinZip Computing, Inc. at http://www.winzip.com to a directory on your hard disk where your program files reside (e.g. *C:\Program Files\PDF Tools AG*).
3. Check the appropriate option to preserve file paths (folder names). The unzip process now creates the following subdirectories:

   bin: Contains the runtime executable binary code.

   doc: Contains documentation files.

   There is the option to download the software as MSI file, which makes the installation easier.
4. To easily use the 3-Heights™ PDF Security Shell from a shell, the directory needs to be included in the "Path" environment variable.
5. Optionally register your license key using the License Manager.
6. If you want to sign documents, proceed with setting up your cryptographic provider as described in chapter Cryptographic Provider.
7. If you want to stamp text, proceed with setting the fonts required as described in chapter Fonts.

2.2 Unix

This section describes installation steps required on all Unix platforms, which includes Linux, Mac OS X, Sun Solaris, IBM AIX, HP-UX, FreeBSD and others.

All Unix Platforms

1. Unpack the archive in an installation directory, e.g. */opt/pdf-tools.com/*
2. Copy or link the executable into one of the standard executable directories, e.g:
   `ln -s /opt/pdf-tools.com/bin/pdfsecure /usr/bin`
3. Verify that the GNU shared libraries required by the product are available on your system:
   - *On Linux*: `ldd pdfsecure`
   - *On AIX*: `dump -H pdfsecure`

   In case you have not installed the GNU shared libraries yet, proceed as follows:
   (a) Go to http://www.pdf-tools.com and navigate to "Support" → "Resouces".
   (b) Download the GNU shared libraries for your platform.
   (c) Extract the archive and copy or link the libraries into your library directory, e.g */usr/lib* or */usr/lib64*.
   (d) Verify that the GNU shared libraries required by the product are available on your system now.
4. Optionally register your license key using the Command Line License Manager Tool.
5. If you want to sign documents, proceed with setting up your cryptographic provider as described in chapter Cryptographic Provider.
6. If you want to stamp text, proceed with setting the fonts required as described in chapter Fonts.

How to set the Environment Variable "Path"

To set the environment variable "Path" on Windows, go to Start ->Control Panel (classic view) ->System ->Advanced ->Environment Variables.

Select "Path" and Edit, then add the directory where *pdfsecure.exe* is located to the "Path". If the environment variable "Path" does not exist, create it.

2.3 Note about the Evaluation Version

The evaluation versions of the 3-Heights™ products automatically add a watermark to the output files.

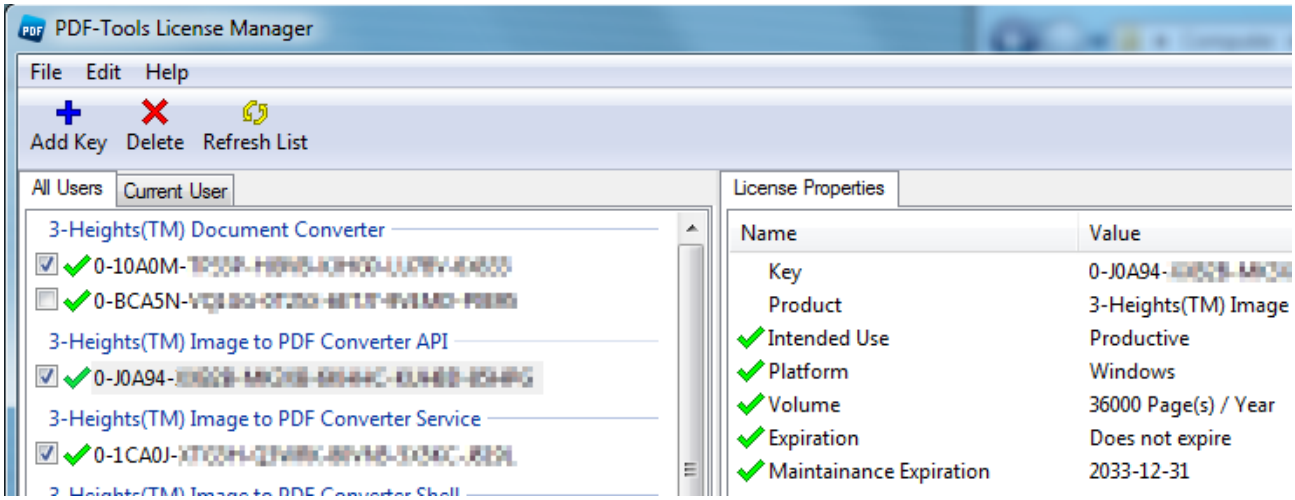
3 License Management

There are three possibilities to pass the license key to the application:

1. The license key is installed using the GUI tool (Graphical user interface). This is the easiest way if the licenses are managed manually. It is only available on Windows.
2. The license key is installed using the shell tool. This is the preferred solution for all non-Windows systems and for automated license management.
3. The license key is passed to the application at runtime via the switch `-lk`. This is the preferred solution for OEM scenarios.

3.1 Graphical License Manager Tool

The GUI tool *LicenseManager.exe* is located in the *bin* directory of the product kit.



List all installed license keys

The license manager always shows a list of all installed license keys in the left pane of the window. This includes licenses of other PDF Tools products. The user can choose between:

- Licenses available for all users. Administrator rights are needed for modifications.
- Licenses available for the current user only.

Add and delete license keys

License keys can be added or deleted with the “Add Key” and “Delete” buttons in the toolbar.

- The “Add key” button installs the license key into the currently selected list.
- The “Delete” button deletes the currently selected license keys.

Display the properties of a license

If a license is selected in the license list, its properties are displayed in the right pane of the window.

Select between different license keys for a single product

More than one license key can be installed for a specific product. The checkbox on the left side in the license list marks the currently active license key.

3.2 Command Line License Manager Tool

The command line license manager tool `licmgr` is available in the bin directory for all platforms except Windows.

A complete description of all commands and options can be obtained by running the program without parameters:

```
licmgr
```

List all installed license keys

```
licmgr list
```

Add and delete license keys

Install new license key:

```
licmgr store X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Delete old license key:

```
licmgr delete X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Both commands have the optional argument `-s` that defines the scope of the action:

- `g`: For all users
- `u`: Current user

Select between different license keys for a single product

```
licmgr select X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.3 License Key Storage

Depending on the platform the license management system uses different stores for the license keys.

Windows

The license keys are stored in the registry:

- `HKLM\Software\PDF Tools AG` (for all users)
- `HKCU\Software\PDF Tools AG` (for the current user)

Mac OS X

The license keys are stored in the file system:

- `/Library/Application Support/PDF Tools AG` (for all users)
- `~/Library/Application Support/PDF Tools AG` (for the current user)

Unix/Linux

The license keys are stored in the file system:

- `/etc/opt/pdf-tools` (for all users)
- `~/.pdf-tools` (for the current user)

Note: The user, group and permissions of those directories are set explicitly by the license manager tool. It may be necessary to change permissions to make the licenses readable for all users. Example:

```
chmod -R go+rx /etc/opt/pdf-tools
```

4 Getting Started

4.1 Basics

Usage

The usage of the 3-Heights™ PDF Security Shell is:

```
pdfsecure [options] input.pdf output.pdf
```

A simple command to encrypt a document requires three parameters: The name of the PDF input file, the PDF output file, and the owner password.

Example: Read the input document "input.pdf", create a new document "output.pdf", set the owner-password of "output.pdf" to "ownerpassword" and the permissions to "allow print" and "allow filling of form fields".

```
pdfsecure -o ownerpassword -p pf input.pdf output.pdf
```

In order to list all available features type "pdfsecure" without any parameters.

4.2 General Settings

-lk: Set License Key

Pass a license key to the application at runtime instead of installing it on the system.

```
pdfsecure -lk X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX input.pdf output.pdf
```

This is only required in an OEM scenario.

4.3 Specify the Folder of the Output File

The output folder can simply be added in front of the output file name

```
pdfsecure input.pdf myfolder\output.pdf
```

or absolute (Windows):

```
pdfsecure input.pdf C:\myfolder\output.pdf
```

4.4 Encryption

Encryption and how it works in PDF

A PDF document can be encrypted to protect its contents from unauthorized access. The encryption process applies encryption to all streams (e.g. images) and strings, but not to other items in the PDF document. This means the structure of the PDF document is accessible, but the content of its pages is encrypted.

When encryption is used in PDF, a security handler must be selected. The 3-Heights™ PDF Security Shell always uses the standard security handler which, according to the PDF Specification, has to be supported by any software that can process encrypted PDF documents.

For more detailed information about PDF encryption in general, see PDF Reference, chapter 3.5.

Owner Password and User Password

The standard security handler allows access permissions and up to two passwords to be specified for a document: An owner password and a user password.

The *user password* protects the document against unauthorized opening and reading. If a PDF document is protected by a user password, either the user or owner password must be provided to open and read the document. If a document has a user password, it must have an owner password as well. If no owner password is defined, the owner password is the same as the user password.

The *owner password* is also referred to as the author's password. This password grants full access to the document. Not only can the document be opened and read, it also allows for changing the document's security settings (access permission and passwords).

The following table shows the four possible combinations of passwords and how an application processing such a PDF document behaves.

Table: Owner and User Passwords

UserPwd	OwnerPwd	Behavior
`none`	`none`	Everyone can read. Everyone can change security settings. (No encryption)
`none`	`set`	Everyone can read. The user password is an empty string. Owner password required to change security settings.
`set`	`none`	User password required to read. The owner password is equal to the user password. User password required to change security settings.
`set`	`set`	User or owner password required to read. Owner password required to change security settings.

Permission Flags

What operations in a PDF document are granted is controlled via its permission flags. In order to set permission flags, the PDF document must be encrypted and have an owner password. The owner password is required to initially set or later change the permission flags.

These access permission flags are:

- Modifying the content of the document
- Copying or extracting text and graphics from the document
- Adding or modifying text annotations and interactive form fields
- Printing the document (low or high quality)
- Filling in form and digitally signing the document
- Assembling the document (inserting, rotating, deleting pages, etc.)

How to Encrypt a PDF Document

If either of the passwords or permission flags is set, the document is encrypted.

If only a user password is set, but no owner password and no permission flags, the owner password is equal to the user password and all permissions are granted.

Example: Create a document where only low resolution printing is allowed.

```
pdfsecure -o ownerpassword -p p input.pdf output.pdf
```

Example: Create a document where only low resolution printing is allowed and the user is prompted for a password upon opening the document. The user must therefore know either the user or the owner password.

```
pdfsecure -o ownerpassword -u userpassword -p p input.pdf output.pdf
```

Example: To create a non-encrypted document, do not set any of the switches -o, -u, -p.

```
pdfsecure input.pdf output.pdf
```

How to Read an Encrypted PDF Document

A PDF document which is not encrypted or protected with an owner password only, can be read and decrypted by the 3-Heights™ PDF Security Shell without providing a password.

A PDF document which is protected by a user password can only be opened if either the user or the owner password is provided using the option -pw. Technically it does not matter later on which of the two passwords was provided, because both will grant full access to the document. However it is up to the application programmer to distinguish between input-documents that are password protected or not.

How secure is PDF Encryption?

Any PDF application that is to process or display a PDF document must be able to read and decrypt the contents of the pages in order to be able to display them. It technically cannot display an encrypted text or image without first decrypting it. A PDF application program has therefore full access to any PDF document it can decrypt and display.

PDF application programs, such as all products of the PDF Security Shell family, or Adobe Acrobat, can open and decrypt PDF documents which have an owner password but no user password, without knowing that password. Otherwise they couldn't display the document. The application at that point has full access to the document. However this does not imply the user of this application is given the same access rights. The user should only be given the access permissions defined by the permission flags and the password he provided. Any PDF application which behaves different from that can allow for changing the security settings or completely removing encryption from the document as long as the original document does not have a user password.

The user password protects the document, so that it only can be opened if the user or owner password is known. No PDF application program can open a user-password protected PDF document without providing the password. The security of such a document however strongly depends on the password itself. Like in most password related situations insecure passwords can easily be found programmatically. E.g. a brute force attempt testing all passwords which either exist as word in a dictionary or have less than six characters only takes minutes.

How to Set Permission Flags equally to Acrobat

In Acrobat 7, there are four different fields/check boxes that can be set. In brackets is the corresponding setting using the parameter -p.

1. Printing Allowed
   - None (`""`)
   - Low Resolution (`p`)
   - High Resolution (`pd`)
2. Changes allowed:
   - None (`""`)
   - Inserting, deleting and rotating pages (`a`)
   - Filling in form fields and signing existing signature fields (`f`)
   - Commenting, filling in form fields, and signing existing signature fields (`fo`)
   - Any except extracting pages (`fom`)
3. Enable copying of text, images and other content (`sc`)
4. Enable text access for screen reader devices for the visually impaired (`s`)

These flags can be combined.

Example: In order to grant permission which are equal to Acrobat's 7 "Printing Allowed: High Resolution" and "Enable copying of text, images and other content", set the flags pdsc 4+16+512+2048=2580.

```
pdfsecure -o ownerpassword -p pdsc input.pdf output.pdf
```

4.5 Cryptographic Provider

In order to use the 3-Heights™ PDF Security Shell's cryptographic functions such as creating or validating digital signatures, a cryptographic provider is required. The cryptographic provider manages certificates, their private keys and implements cryptographic algorithms.

The 3-Heights™ PDF Security Shell can use various different cryptographic providers. The following list shows, for which type of signing certificate which provider can be used.

USB Token or Smart Card

These devices typically offer a PKCS#11 interface, which is the recommended way to use the certificate → PKCS#11 Provider.

On Windows, the certificate is usually also available in the Microsoft CryptoAPI. This provider is not recommended, unless you experience problems with your device's PKCS#11 interface.

If you need to sign documents on a non-Windows system with an USB token that does not come with middleware for your platform, you can use the 3-Heights™ Signature Creation and Validation Service.

If you need to sign documents on Windows in a non-interactive or locked session[1], use the 3-Heights™ Signature Creation and Validation Service.

Hardware Security Module (HSM)

HSMs always offer very good PKCS#11 support → PKCS#11 Provider

For more information and installation instructions see separate document TechNotePKCS11.pdf.

Soft Certificate

Soft certificates are typically PKCS#12 files that have the extension `.pfx` or `.p12` and contain the signing certificate as well as the private key and trust chain (issuer certificates). Soft certificate files cannot be used directly. Instead, they must be imported into the certificate store of a cryptographic provider.

- *All Platforms*: The recommended way of using soft certificates is to import them into a store that offers a PKCS#11 interface and use the PKCS#11 Provider. For example:
  - A HSM
  - openCryptoki on Linux
  - PKCS#11 softtoken on Solaris

  For more information and installation instructions of the above stores see separate document TechNotePKCS11.pdf.
- *Windows*: If no PKCS#11 provider is available, soft certificates can be imported into Windows certificate store, which can then be used as cryptographic provider → Microsoft CryptoAPI

Signature Service

Signature services are a convenient alternative to storing certificates and key material locally. The 3-Heights™ PDF Security Shell can use various different services whose configuration is explained in the following sections of this documentation:

- 3-Heights™ Signature Creation and Validation Service
- SwissSign Personal Signing Service
- SwissSign SuisseID Signing Service
- QuoVadis sealsign
- Swisscom All-in Signing Service

4.5.1 PKCS#11 Provider

PKCS#11 is a standard interface offered by most cryptographic devices such as HSMs, USB Tokens or sometimes even soft stores (e.g. openCryptoki).

More information on and installation instructions of the PKCS#11 provider of various cryptographic devices can be found in the separate document TechNotePKCS11.pdf.

[1] See the description of the 3-Heights™ Signature Creation and Validation Service for more details on this topic.

Configuration

Provider Option -cp

The provider configuration string has the following syntax:

`"PathToDll;SlotId;Pin"`

- `PathToDll` is the path to driver library filename, which is provided by the manufacturer of the HSM, UBS token or smart card. Examples:
  - The SuisseID USB Tokens use *cvp11.dll*
  - The CardOS API from Atos (Siemens) uses *siecap11.dll*
  - The IBM 4758 cryptographic coprocessor uses *cryptoki.dll*
  - Devices from Aladdin Ltd. use *etpkcs11.dll*
- `SlotId` is optional, if it is not defined, it is searched for the first slot that contains a running token.
- `Pin` is optional, if it is not defined, the submission for the pin is activated via the pad of the token. If this is not supported by the token, the following error message is raised when signing: "Cannot access private key".

Examples:

- `Provider = "\WINDOWS\system32\siecap11.dll;4;123456"`

Interoperability Support

The following cryptographic token interface (PKCS#11) products have been successfully tested:

- SafeNet Protect Server
- SafeNet Luna
- SafeNetAuthentication Client
- IBM OpenCrypTokI
- CryptoVision
- Siemens CardOS

Selecting a Certificate for Signing

The 3-Heights™ PDF Security Shell offers different ways to select a certificate. The product tries the first of the following selection strategies, for which the required values have been specified by the user.

1. **Certificate fingerprint**
   Option -cfp
   - SHA1 fingerprint of the certificate. The fingerprint is 20 bytes long and can be specified in hexadecimal string representation, e.g. "b5 e4 5c 98 5a 7e 05 ff f4 c6 a3 45 13 48 0b c6 9d e4 5d f5". In Windows certificate store this is called "Thumbprint", if "Thumbprint algorithm" is "sha1".
2. **Certificate Issuer and SerialNumber**
   Options -ci and -cno
   - Certificate Issuer (e.g. "QV Schweiz CA"), in Windows certificate store this is called "Issued By".
   - Serial number of the certificate (hexadecimal string representation, e.g. "4c 05 58 fb"). This is a unique number assigned to the certificate by its issuer. In Windows certificate store this is the field called "Serial number" in the certificate's "Details" tab.
3. **Certificate Name and optionally Issuer**
   Options -cn and -ci
   - Common Name of the certificate (e.g. "PDF Tools AG"), in Windows certificate store this is called "Issued To".
   - Optional: Certificate Issuer (e.g. "QV Schweiz CA"), in Windows certificate store this is called "Issued By".

Using PKCS#11 stores with missing issuer certificates

Some PKCS#11 devices contain the signing certificate only. However, in order to embed revocation information it is important, that the issuer certificates, i.e. the whole trust chain, is available as well.

On Windows, missing issuer certificates can be loaded from the Windows certificate store. So the missing certificates can be installed as follows:

1. Get the certificates of the trust chain. You can download them from the website of your certificate provider or do the following:

(a) Sign a document and open the output in Adobe Acrobat
(b) Go to “Signature Properties” and then view the signer’s certificate
(c) Select a certificate of the trust chain
(d) Export the certificate as “Certificate File” (extension .cer)
(e) Do this for all certificates of the trust chain

2. Open the exported files by double clicking on them in the Windows Explorer
3. Click button “Install Certificate...”
4. Select “automatically select the certificate store based on the type of certificate” and finish import

4.5.2 Microsoft CryptoAPI Provider

Microsoft CryptoAPI (MS-CAPI, CAPI) offers access to the certificates stored in the Windows certificate store and other devices, such as USB tokens, with Windows integration. Microsoft CryptoAPI does not support some new cryptographic algorithms. Therefore it is recommended to use the PKCS#11 Provider if possible.

Configuration

Provider Option -cp

The provider configuration string has the following syntax:

```
"[ProviderType:]Provider[;PIN]"
```

The ProviderType and PIN are optional. The corresponding drivers must be installed on Windows.

Examples:

- `Provider = "Microsoft Base Cryptographic Provider v1.0"`
- `Provider = "Microsoft Strong Cryptographic Provider"`
- `Provider = "PROV_RSA_AES:Microsoft Enhanced RSA and AES Cryptographic Provider"`

The provider type PROV_RSA_AES supports the SHA-2 hash algorithms for signature validation (not signature creation). This provider type is recommended in order to validate signatures if no PKCS#11 device is available.

Optionally, when using an advanced certificate, the pin code can be passed as an additional, semi-column separated parameter. This does not work with qualified certificates, because they always require the pin code to be entered manually and every time. If the name of the provider is omitted, the default provider is used.

Examples, “123456” being the pin code:

- `Provider = "Microsoft Base Cryptographic Provider v1.0;123456"`
- `Provider = ";123456"`

Certificate Store Option -csn

The value for the certificate store depends on the OS. Supported values are: “CA”, “MY” and “ROOT”. For signature creation the default store “MY” is usually the right choice.

Store Location Option -csl

Either of the following store locations

- Local Machine
- Current User (default)

Usually personal certificates are stored in the current user location and company-wide certificates are stored under local machine.

The current user’s store is only available, if the user profile has been loaded. This may not be the case in certain environments such as within an IIS web application or COM+ applications. Use the store of the Local Machine, if the user profile cannot be loaded.

Certificates in the store Local Machine are available to all users. However, in order to sign a document, you need access to the signing certificate’s private key. The private key is protected by Windows ACLs and typically readable for Administrators only. Use the Microsoft Management Console (mmc.exe) in order to grant access to the private key for other users as follows: Add the Certificates Snap-in for the certificates on Local Machine. Right-click on the signing certificate, click on “All Tasks” and then “Manage Private Keys...” where you can set the permissions.

Selecting a Certificate for Signing

First, the certificate store defined by the provider is used. Within the store the selection of the signing certificate works the same as with the PKCS#11 provider, which is described here: Selecting a Certificate for Signing

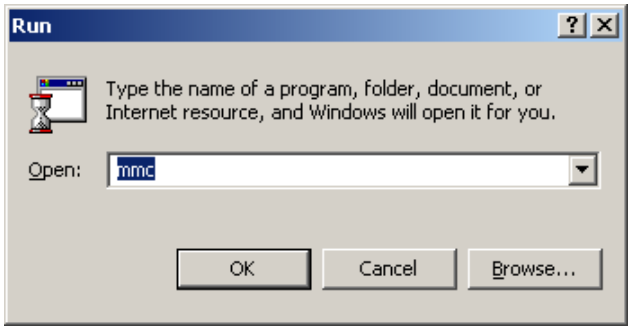
Certificates

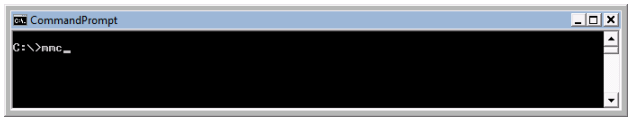
In order to sign a PDF document, a valid, existing certificate name must be provided and its private key must be available.

There are various ways to create or obtain a certificate. How this is done is not described in this document. This document describes the requirements for, and how to use the certificate.

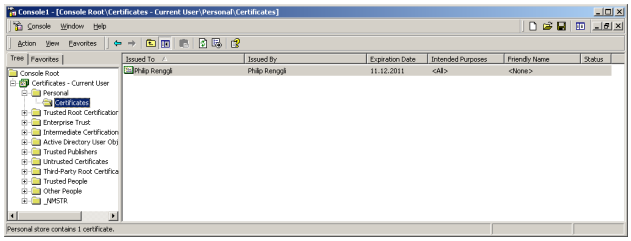
On the Windows operating system certificates can be listed by the Microsoft Management Console (MMC), which is provided by Windows. In order to see the certificates available on the system, do the following steps:

1. To launch the MMC, go to Start ->Run...->type “mmc”, or start a Command Prompt and type “mmc”.

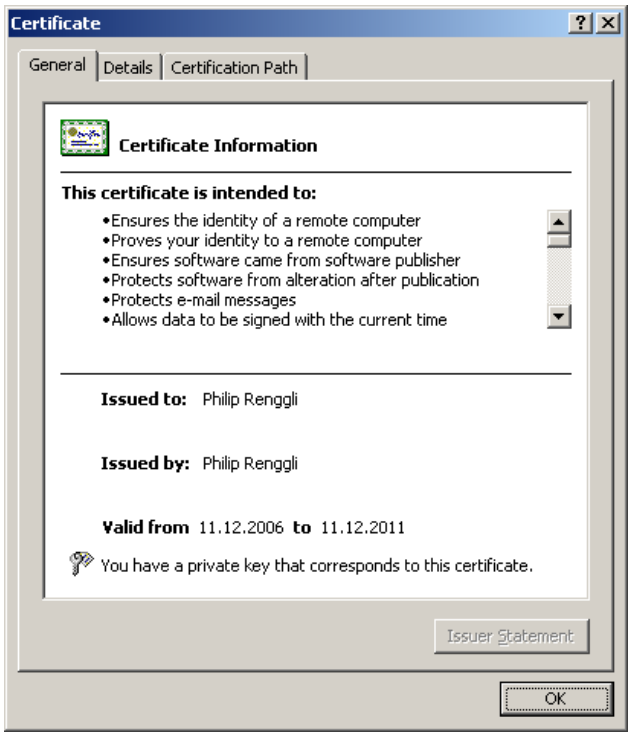




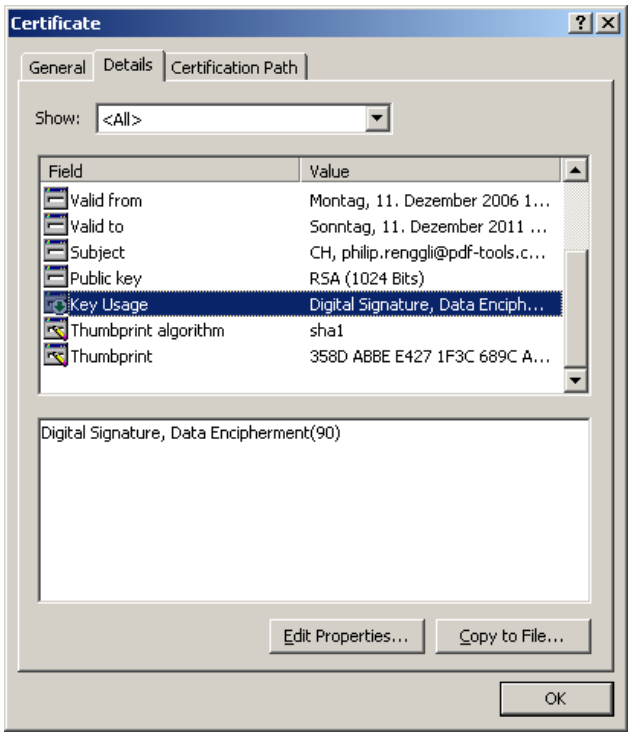
2. Under “File” ->“Add/Remove Snap-in”
3. Choose “Certificates” and click the “Add” button
4. In the next window choose to manage certificates for “My user account”
5. Click “Finish”
6. The certificate must be listed under the root “Certificates - Current User”, for example as shown in the screenshot below:



7. Double-click the certificate to open. The certificate name corresponds to the value “Issued to:”.

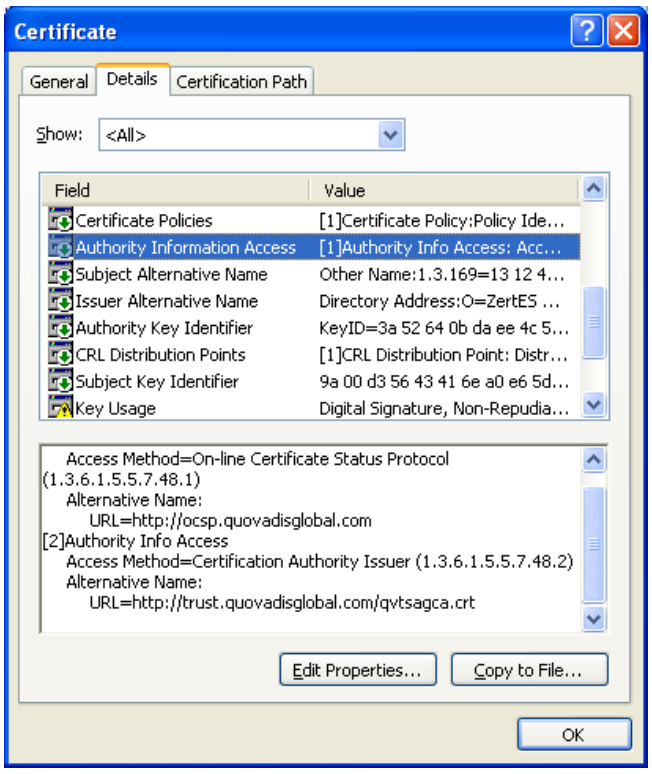

Certificate
General
Details
Certification Path
Certificate Information
This certificate is intended to:
•Ensures the identity of a remote computer
•Proves your identity to a remote computer
•Ensures software came from software publisher
•Protects software from alteration after publication
•Protects e-mail messages
•Allows data to be signed with the current time
Issued to: Philip Renggli
Issued by: Philip Renggli
Valid from 11.12.2006 to 11.12.2011
You have a private key that corresponds to this certificate.
Issuer Statement
OK

8. In the tab Detail of the certificate, there is a field named “Key Usage”. This field must contain the value “Digital Signature”. Additional values are optional, see also screenshot. You must have the private key that corresponds to this certificate.



Qualified Certificates

A qualified certificate can be obtained from a certificate authority (CA). Besides the requirements listed in the previous chapter it has the additional requirement to contain the key “Authority Information Access” which contains the information about the OCSP server.



4.5.3 3-Heights™ Signature Creation and Validation Service

The 3-Heights™ Signature Creation and Validation Service provides HTTP protocol based remote access to cryptographic providers such as smartcards, USB tokens, and other cryptographic infrastructure such as HSMs.

Use of the 3-Heights™ Signature Creation and Validation Service provides the following advantages:

1. By means of this service the tokens can be hosted centrally and used by any client computer which has access to the service.
2. Cryptographic devices that can be used on Windows only can be made accessible to siging processes running on Non-Windows systems.
3. Cryptographic devices can be made accessible to processes running in non-interactive sessions. Many cryptographic devices must always be used in an interactive session for two reasons. First, the middleware requires the user to enter the pin interactively to create a qualified electronic signature. Second, USB tokens and smart cards are managed by Windows such that the device is available only to the user currently using the computer’s console. Therefore, services, remotely logged in users and applications running in locked sessions have no access to the device.

Note that this is a separate product and this chapter describes its usage with the 3-Heights™ PDF Security Shell only. For more information on the 3-Heights™ Signature Creation and Validation Service and installation instructions, please refer to its separate user manual.

Configuration

Provider Option -cp

The provider configuration string has the following syntax:

`"http://server.mydomain.com:<port>/<token>;<password>"`

Where:

- `server.mydomain.com` is the hostname of the server
- `<port>` is optional, port of the server.
- `<token>` the ID of the token.

- `<password>` password of the token.

Examples:

- `Provider = "http://server.mydomain.com:8080/0001;pass01"`

A more detailed description can be found in the user manual of the 3-Heights™ Signature Creation and Validation Service.

Selecting a Certificate for Signing

Selection of the signing certificate works the same as if the token was used directly: Selecting a Certificate for Signing.

4.5.4 SwissSign Personal Signing Service

Provider Option -cp

The provider configuration string contains the URL to the service endpoint.

Provider Configuration

The provider can be configured using provider session properties.

There are two types of properties:

- *String Properties:*
  String properties are set using option -cps.
- *File Properties:*
  File properties are set using option -cpf.

Name	Type	Required	Value
Identity	String	required	The identity of your signing certificate. Example: `My Company:Signing Cert 1`
DSSProfile	String	required	`http://dss.swisssign.net/dss/profile/pades/1.0`
SSLClientCertificate	File	required	SSL client certificate in PKCS#12 Format (.p12, .pfx). File must contain the certificate itself, all certificates of the trust chain and the private key.
SSLClientCertificatePassword	String	optional	Password to decrypt the private key of the SLL client certificate.
SSLServerCertificate	File	recommended	Certificate of the server or its issuer (CA) certificate in DER Format (.der, .cer) Note: If this property is not set, the server certificate is not verified at all!
RequestID	String	recommended	Any string that can be used to track the request. Example: An UUID like `AE57F021-C0EB-4AE0-8E5E-67FB93E5BC7F`

Signature Configuration

The signature can be customized using standard options of the 3-Heights™ PDF Security Shell.

Description	Required	Value	Setting
Common Name	required	The name of the signer should be set[2].	Option -cn.
Time-stamp	optional	`"urn:ietf:rfc:3161"`	Option -tsu
Revocation Info	optional	`true` to embed OCSP responses or CRL.	Option -co
Visual Appearance	optional	See separate chapter on creating a visual appearance.	

Proxy Configuration

If a proxy is used for the connection to the service, see chapter How to Use a Proxy for more information.

[2]This parameter is not used for certificate selection, but for the signature appearance and signature description in the PDF only.

4.5.5 SwissSign SuisseID Signing Service

In order to use the SuisseID Signing Service, please contact Swiss Post Solutions AG (suisseid@post.ch) to obtain access credentials. Prior to invoking the SuisseID Signing Service, user authentication via the SuisseID Identity Provider (IDP) is a pre-requisite. So the calling application must integrate via SAML (e.g. SuisseID SDK) with the SuisseID Identity Provider. The IDP issues SAML tokens upon successful user authentication.

Note that the name of the signature should be the signer's name (e.g. "<givenname> <surname>"). The signer's name can be retrieved for the SAML token as the IDP provides this as qualified attributes (yellowid verified).

Provider Option -cp

The provider configuration string contains the URL to the service Endpoint.

Provider Configuration

The provider can be configured using provider session properties.

There are two types of properties:

- *String Properties:*
  String properties are set using option -cps.
- *File Properties:*
  File properties are set using option -cpf.

Name	Type	Required	Value
SAMLToken	File	required	SAML token issued by the SuisseID Identity Provider (IDP). Example: `C:\temp\my-saml.xml`
SSLClientCertificate	File	required	SSL client certificate in PKCS#12 Format (.p12, .pfx). File must contain the certificate itself, all certificates of the trust chain and the private key.
SSLClientCertificatePassword	String	optional	Password to decrypt the private key of the SLL client certificate.
SSLServerCertificate	File	recommended	Certificate of the server or its issuer (CA) certificate in DER Format (.der, .cer) Note: If this property is not set, the server certificate is not verified at all!

Signature Configuration

The signature can be customized using standard options.

Description	Required	Value	Setting
Common Name	required	The name of the signer should be set[3].	Option -cn.
Time-stamp	optional	`"urn:ietf:rfc:3161"`	Option -tsu
Revocation Info	optional	`true` to embed OCSP responses or CRL.	Option -co
Visual Appearance	optional	See separate chapter on creating a visual appearance.	

Proxy Configuration

If a proxy is used for the connection to the service, see chapter How to Use a Proxy for more information.

[3]This parameter is not used for certificate selection, but for the signature appearance and signature description in the PDF only.

4.5.6 QuoVadis sealsign

Provider Option -cp

The provider configuration string contains the URL to the QuoVadis sealsign service. For example:
`"http://services.sealsignportal.com:18080/sealsign/ws/BrokerClient"`

Provider Configuration

The provider can be configured using provider session properties that can be set using the options -cps or -cpf.

Name	Type	Required	Value
Identity	String	required	The account ID is the unique name of the account specified on the server. Example: `Rigora`
Profile	String	required	The profile identifies the signature specifications by a unique name. Example: `Default`
secret	String	required	The secret is the password which secures the access to the account. Example: `NeE=EKEd33FeCk70`
clientId	String	optional	A client ID can be used to help separating access and creating better statistics. If specified in the account configuration it is necessary to provide this value. Example: `3949-4929-3179-2818`
pin	String	required	The PIN code is required to activate the signing key. Example: `123456`
MessageDigestAlgorithm	String	optional	The message digest algorithm to use. Note that the supported algorithms depend on the provider. Default: `SHA-256` Alternatives: `SHA-1`, `SHA-384`, `SHA-512`, `RIPEMD-160`, `RIPEMD-256`

Signature Configuration

The signature can be customized using standard options.

Description	Required	Value	Setting
Common Name	required	The name of the signer should be set[4].	Option -cn.
Time-stamp	-	Not available.	
Revocation Info	optional	`true` to embed OCSP responses or CRL.	Option -co
Visual Appearance	optional	See separate chapter on creating a visual appearance.	

Proxy Configuration

If a proxy is used for the connection to the service, see chapter How to Use a Proxy for more information.

[4]This parameter is not used for certificate selection, but for the signature appearance and signature description in the PDF only.

4.5.7 Swisscom All-in Signing Service

General Properties

To use the signature service, the following general properties have to be set:

Description	Required	Value	Setting
Common Name	required	Name of the signer[5].	Option -cn
Provider	required	The service endpoint URL of the REST service. Example: `https://ais.swisscom.com/AIS-Server/rs/v1.0/sign`	Option -cp
Time-stamp	optional	`"urn:ietf:rfc:3161"`	Option -tsu
Revocation Info	optional	`true` to embed OCSP responses	Option -co

If a proxy is used for the connection to the service, see chapter How to Use a Proxy for more information.

Provider Session Properties

In addition to the general properties, a few provider specific session properties have to be set.

There are two types of properties:

- *String Properties:*
  String properties are set using option -cps.
- *File Properties:*
  File properties are set using option -cpf.

Name	Type	Required	Value
DSSProfile	String	required	`http://ais.swisscom.ch/1.0`
SSLClientCertificate	File	required	SSL client certificate in PKCS#12 Format (.p12, .pfx). File must contain the certificate itself, all certificates of the trust chain and the private key.
SSLClientCertificatePassword	String	optional	Password to decrypt the private key of the SLL client certificate.
SSLServerCertificate	File	recommended	Certificate of the server or its issuer (CA) certificate in DER Format (.der, .cer) Note: If this property is not set, the server certificate is not verified at all!
Identity	String	required	The identity string as provided by Swisscom: `<customer name>:<key identity>`
RequestID	String	recommended	Any string that can be used to track the request. Example: An UUID like `AE57F021-C0EB-4AE0-8E5E-67FB93E5BC7F`

[5]This parameter is not used for certificate selection, but for the signature appearance and signature description in the PDF only.

On-Demand Certificates

To request an on-demand certificate, the following additional property has to be set:

Name	Type	Required	Value
SwisscomAllInOnDemandDN	String	required	The requested distinguished name. Example: `cn=Hans Muster,o=ACME,c=CH`

Step-Up Authorization using Mobile-ID

To use the step-up authorization, the following additional properties have to be set:

Name	Type	Required	Value
SwisscomAllInMSISDN	String	required	Mobile phone number. Example: `+41798765432`
SwisscomAllInMessage	String	required	The message to be displayed on the mobile phone. Example: `Pipapo halolu.`
SwisscomAllInLanguage	String	required	The language of the message. Example: `DE`

Those properties have to comply with the Swisscom Mobile-ID specification.

4.6 How to Create Digital Signatures

This chapter describes the steps that are required to create different types of digital signatures. A good introductory example can be found in the chapter Digital Signatures.

4.6.1 How to Create a PAdES LTV Signature

In order to create a PAdES LTV signature, the following is required:

1. *An advanced or qualified signing certificate.*
   For requirements and preparation steps see the sample in chapter Digital Signatures. Make sure the store of your cryptographic provider contains all certificates of the trust chain, including the root certificate.
2. *Embed revocation information.*
   Do not use the option -co.
3. *Add a Time-stamp.*
   Use the option -tsu.
4. *Proper error handling.*
   A proper error handling is crucial in order to ensure the creation of correctly signed documents. The output document was signed successfully, if and only if the 3-Heights™ PDF Security Shell returns code 0 (success).

4.6.2 How to Create a Time-stamp Signature

For a Time-stamp signature no local signing certificate is required. Instead the Time-stamp signature requested from the Time-stamp Authority (TSA) is embedded into the document.

Example: Create a Time-stamp signature using the option -dts.

```
pdfsecure ^
-cp "cvp11.dll"
-tsu "http://server.mydomain.com:80/tsa" ^
-dts ^
input.pdf output.pdf
```

A Cryptographic Provider is required on non-Windows systems only.

4.6.3 How to Create a Visual Appearance of a Signature

Each signature may have a visual appearance on a page of the document. The visual appearance is optional and has no effect on the validity of the signature. Because of this and because a visual appearance may cover important content of the page, many applications choose to create an invisible signature.

By default, the 3-Heights™ PDF Security Shell creates an appearance in the lower left corner of the last page, which looks as shown below:

How to Create an Invisible Signature

Invisible signatures have no visual appearance and can be created by setting an empty rectangle using the Option -ar:

```
-ar 0 0 0 0
```

How to Create a Visual Appearance

Different properties of the visual appearance can be specified.

Page and Position
: See options -ap and -ar.

Color
: See options -acf and -acs.

Line Width
: The line width of the background rectangle, see option -al.

Text
: Two text fragments can be set using two different fonts and font sizes, see options -at1, -at2, -af1 and -af2.

Background image
: See options -abg.

4.6.4 Miscellaneous

Caching of CRLs, OCSP and TSP Reponses

In order to improve the speed when mass signing, the 3-Heights™ PDF Security Shell provides a caching algorithm to store CRL (Certificate Revocation List), OCSP (Online Certificate Status Protocol), TSP (Time-stamp Protocol) and data from signature services. This data is usually valid over period of time that is defined by the protocol, which is normally at least 24 hours. Caching improves the speed, because there are situations when the server does not need to be contacted for every digital signature. The following caches are stored automatically by the 3-Heights™ PDF Security Shell at the indicated locations within the directory for temporary files:

OCSP responses:	*ocsp/server-hash.der*
CRL:	*crl/server.der*
TSP responses[6]	*tsp/server.der*
Service data:	*sig/hash.bin*

The caches can be cleared by deleting the files. Usage of the caches can be deactivated by setting the option -nc. The files are updated if the current date and time exceeds the “next update” field in the OCSP or CRL response respectively or the cached data was downloaded more than 24 hours ago.

[6] The sizes of the TSP responses are cached only. Cached TSP responses cannot be embedded but used for the computation of the signature length only.

The directory for temporary files is determined as follows. The product checks for the existence of environment variables in the following order and uses the first path found:

- *Windows:*
  1. The path specified by the *TMP* environment variable.
  2. The path specified by the *TEMP* environment variable.
  3. The path specified by the *USERPROFILE* environment variable.
  4. The Windows directory.
- *Unix:*
  1. The path specified by the *PDFTMPDIR* environment variable.
  2. The path specified by the *TMP* environment variable.
  3. The */tmp* directory.

How to Use a Proxy

The 3-Heights™ PDF Security Shell can use a proxy server for all communication to remote servers, e.g. to download CRL or for communication to a signature service. The proxy server can be configured using the provider session property *Proxy*. The property's value must be a string with the following syntax:

`http[s]://<user:password>@host:<port>`

Where:

- `http` / `https`: Protocol for connection to proxy.
- `user:password` (optional): Credentials for connection to proxy (basic authorization).
- `host`: Hostname of proxy.
- `port`: Port for connection to proxy.

For SSL connections, e.g. to a signature service, the proxy must allow the HTTP CONNECT request to the signature service.

Example: Configuration of a proxy server that is called `myproxy` and accepts HTTP connections on port `8080`.

```
-cps "Proxy" "http://myproxy:8080"
```

Configuration of Proxy Server and Firewall

For the application of a Time-stamp or online verification of certificates, the signature software requires access to the server of the certificates' issuer (e. g. http://ocsp.quovadisglobal.com or http://platinum-qualified-g2.ocsp.swisssign.net/) via HTTP. The URL for verification is stored in the certificate; the URL for Time-stamp services is provided by the issuer. In case these functions are not configured, no access is required.

In organizations where a web proxy is in used, it must be ensured that the required MIME types are supported. These are:

OCSP

- `application/ocsp-request`
- `application/ocsp-response`

Time-stamp

- `application/timestamp-query`
- `application/timestamp-reply`

Signature services

- Signature service specific MIME types.

4.7 How to Validate Digital Signatures

Validation of a Qualified Electronic Signature

There are basically three items that need to be validated:

1. Trust Chain
2. Revocation Information (optional)
3. Time-stamp (optional)

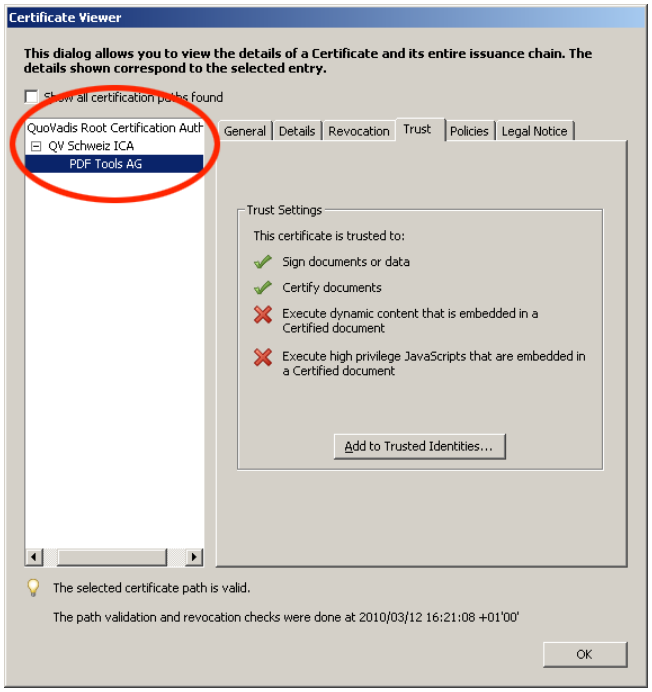
Validation can be in different ways, e.g. Adobe Acrobat, from which the screenshots below are taken.

Trust Chain

Before the trust chain can be validated, ensure the root certificate is trusted. There are different ways to add a certificate as trusted root certificate. The best way on Windows is this:

1. Retrieve a copy of the certificate containing a public key. This can be done be requesting it from the issuer (your CA) or by exporting it from an existing signature to a file (*CertExchange.cer*). Ensure you are not installing a malicious certificate!
2. Add the certificate to the trusted root certificates. If you have the certificate available as file, you can simply double-click it to install it.

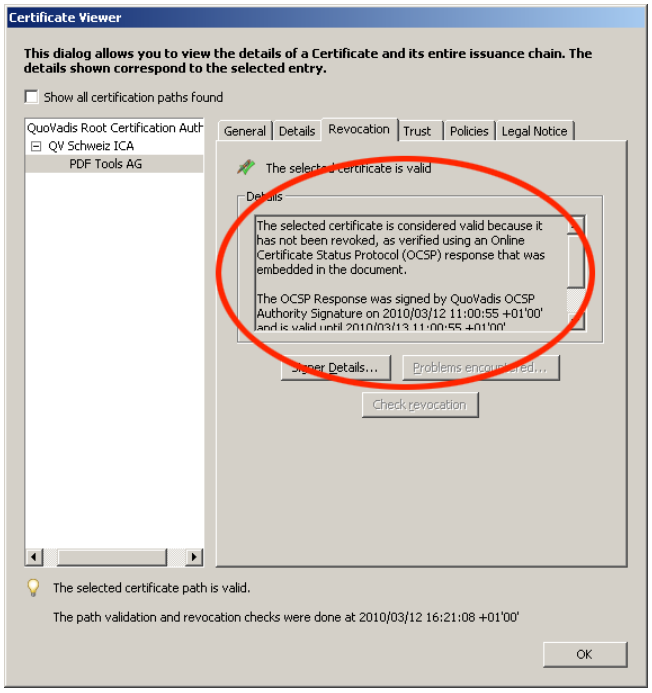
After that you can validate the signature, e.g. by open the PDF document in Adobe Acrobat, right-click the signature and select “Validate”, then select “Properties” and select the tab “Trust”. There the certificate should be trusted to “sign documents or data”.


Certificate Viewer
This dialog allows you to view the details of a Certificate and its entire issuance chain. The details shown correspond to the selected entry.
Show all certification paths found
QuoVadis Root Certification Auth
QV Schweiz ICA
PDF Tools AG
General
Details
Revocation
Trust
Policies
Legal Notice
Trust Settings
This certificate is trusted to:
Sign documents or data
Certify documents
Execute dynamic content that is embedded in a Certified document
Execute high privilege JavaScripts that are embedded in a Certified document
Add to Trusted Identities...
The selected certificate path is valid.
The path validation and revocation checks were done at 2010/03/12 16:21:08 +01'00'
OK

Revocation Information

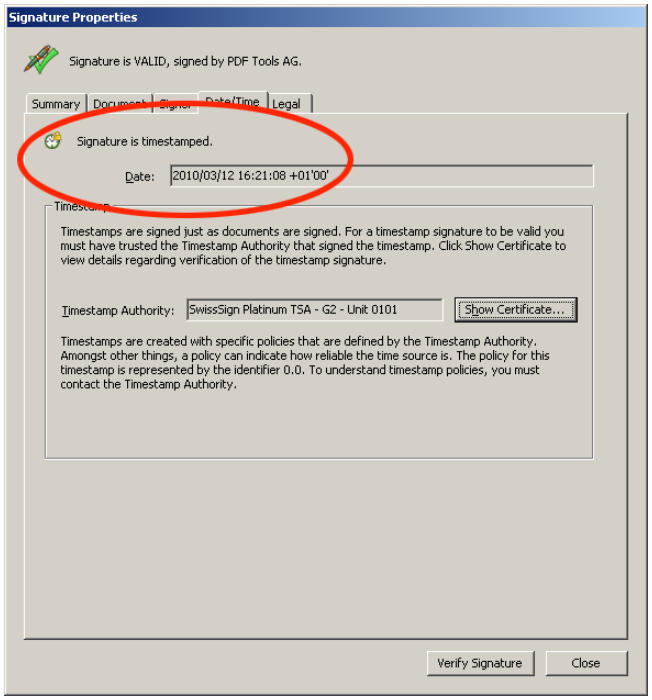
An OCSP response or CRL must be available. This is shown in the tab “Revocation”. The details should mention that “the certificate is considered *valid*”.

The presence of revocation information must be checked for the signing certificate and all certificates of its trust chain except for the root certificate.


Certificate Viewer
This dialog allows you to view the details of a Certificate and its entire issuance chain. The details shown correspond to the selected entry.
Show all certification paths found
QuoVadis Root Certification Auth
QV Schweiz ICA
PDF Tools AG
General
Details
Revocation
Trust
Policies
Legal Notice
The selected certificate is valid
Details
The selected certificate is considered valid because it has not been revoked, as verified using an Online Certificate Status Protocol (OCSP) response that was embedded in the document.
The OCSP Response was signed by QuoVadis OCSP Authority Signature on 2010/03/12 11:00:55 +01'00'
Signer Details...
Problems encountered...
Check revocation
The selected certificate path is valid.
The path validation and revocation checks were done at 2010/03/12 16:21:08 +01'00'
OK

Time-stamp

The signature can optionally contain a Time-stamp. This is shown in the tab “Date/Time”. The certificate of the Time-stamp server must also be trusted, i.e. its trust chain should be validated as described in the section Trust Chain above.


Signature Properties
Signature is VALID, signed by PDF Tools AG.
Summary
Date/Time
Legal
Signature is timestamped.
Date: 2010/03/12 16:21:08 +01'00'
Timestamps are signed just as documents are signed. For a timestamp signature to be valid you must have trusted the Timestamp Authority that signed the timestamp. Click Show Certificate to view details regarding verification of the timestamp signature.
Timestamp Authority: SwissSign Platinum TSA - G2 - Unit 0101
Show Certificate...
Timestamps are created with specific policies that are defined by the Timestamp Authority. Amongst other things, a policy can indicate how reliable the time source is. The policy for this timestamp is represented by the identifier 0.0. To understand timestamp policies, you must contact the Timestamp Authority.
Verify Signature
Close

Validation of a PAdES LTV Signature

Verifying if a signature conforms to the PAdES LTV standard is similar to validating a Qualified Electronic Signature.

The following must be checked:

1. Trust Chain
2. Revocation information
3. Time-stamp
4. LTV expiration date
5. Other PAdES Requirements

Trust Chain

Trust chain validation works the same as for validating Qualified Electronic Signatures.

Revocation information

Revocation information (OCPS response or CRL) must be valid and embedded into the signature. In the details, verify that the revocation check was performed using data that was "was *embedded in the signature*". Revocation information that "was contained in the *local cache*" or "was requested *online*" is not embedded into the signature and does not meet PAdES LTV requirements.

Time-stamp

A Time-stamp must be embedded and validated as described for validating Qualified Electronic Signatures. If a document contains multiple Time-stamps, all but the latest one must contain revocation information.

LTV expiration date

The long term validation ability expires with the expiration of the signing certificate of the latest Time-stamp. The life-time of the protection can be further extended beyond the life-of the last Time-stamp applied by adding further DSS information to validate the previous last Time-stamp along with a new Time-stamp.

Other PAdES Requirements

Certain other PAdES requirements, such as requirements on the PKCS#7 CMS, cannot be validated using Adobe Acrobat. For this, use the 3-Heights™ PDF Security API for validation.

4.8 Fonts

Some features of the 3-Heights™ PDF Security Shell require fonts to be installed, e.g. when stamping text.

The location of the font directories depends on the operating system:

- Windows: `%SystemRoot%\Fonts` and directory `Fonts`, which must be a direct sub-directory of where the main DLL or executable resides.
- Mac: `/System/Library/Fonts` and `/Library/Fonts`
- Unix: `$PDFFONTDIR` or `/usr/lib/X11/fonts/Type1`

The fonts of the font directories and their properties are cached in a font cache, located in the files `font-database*` in the temporary files folder. The cache files have to be removed manually, if fonts are added or removed from these directories.

The directory for temporary files is determined as follows. The product checks for the existence of environment variables in the following order and uses the first path found:

- *Windows:*
  1. The path specified by the *TMP* environment variable.
  2. The path specified by the *TEMP* environment variable.
  3. The path specified by the *USERPROFILE* environment variable.
  4. The Windows directory.
- *Unix:*

1. The path specified by the *PDFTMPDIR* environment variable.
2. The path specified by the *TMP* environment variable.
3. The */tmp* directory.

Example: Command to remove the font cache files on Windows

```
del %TMP%\font-database*
```

5 Reference Manual

Switches are options that are provided with the command to define how the document should be processed.

Switches can occur in two forms: As stand-alone option, such as `-v` (verbose mode) or they may require a parameter, such as `-pw password` (set password to read encrypted input document).

The last two parameters of the command line should always be the input and the output document. Switches are parsed from left to right. If the same switch is applied multiple times the last set value is applied.

5.1 Encryption

-fe: Force Encryption

File encryption is not allowed by the PDF/A standard. Therefore 3-Heights™ PDF Security Shell aborts and returns an error, when encryption is configured and an imput file is PDF/A. Use this option, in order to enable encryption of PDF/A conforming files. The conformance of the output file is downgraded to PDF.

-fr: Set String Crypt Filter

Set the string crypt filter. Supported values are the following strings: “None”, “V2”, “AESV2” and “AESV3”. Setting an empty string selects the default filter.
Crypt filter:

`None` The application does not decrypt data

`V2` (PDF 1.1, default) The application asks the security handler for the encryption key and implicitly decrypts data using the RC4 algorithm.

`AESV2` (PDF 1.6) The application asks the security handler for the encryption key and implicitly decrypts data with using the AES-V2 128 bit algorithm.

`AESV3` (PDF 1.7) The application asks the security handler for the encryption key and implicitly decrypts data with using the AES-V3 256 bit algorithm.

-fm: Set Stream Crypt Filter

Set the stream crypt filter. Supported values are the following strings: “None”, “V2”, “AESV2” and “AESV3”. Note that certain PDF viewers require the stream crypt filter to be equal to the string crypt filter, e.g. both must be RC4 or AES. Other tools, such as the 3-Heights™ PDF Tools do not have this limitation. Setting an empty string selects the default filter.

Example: Set the stream crypt filter and the string crypt file to AESV2

```
pdfsecure -o owner -fm AESV2 -fr AESV2 input.pdf output.pdf
```

-id: Add entries to the info object (metadata)

This switch receives key / value pairs as a parameter. The pair is added as a new entry to the info object. If the entry already exists then the previous entry is overwritten. If the key corresponds to a standard metadata key then the XMP metadata is updated accordingly.

Example: Overwrite the default producer:

```
pdfsecure -id Producer=MyProgram input.pdf output.pdf
```

-k: Set the Length of the Encryption Key

The key length is a determining factor of the strength of the encrypting algorithm and the amount of time to break the cryptographic system. For RC4 the key length can be any value from 40 to 128 that is a multiple of 8. For AESV2 the key length is automatically set to 128, for AESV3 to 256.
Notes:

- Certain PDF viewers only support 40 and 128 bit encryption. Other tools, such as the 3-Heights™ tools also support other encryption key lengths.
- 256 bit encryption requires Acrobat 9 or later.
- If the selected permission flags require a minimum key length, the key length is automatically adjusted (e.g. to 128 bits)

-o: Set the Owner Password

The owner password is required to change the security settings of the document. In order to apply permission flags, an owner password must be set. Permission flags are set with the switch `-p`.

Example: Encrypt a document and set the owner password to “owner”.

```
pdfsecure -o owner input.pdf output.pdf
```

-p: Set the Permission Flags

This option sets the permission flags. It is only usable in combination with encrypted documents, i.e. an owner password must be set. By default all permissions are granted. The permissions that can be granted are listed below.

`p`	allow printing (low resolution)
`m`	allow changing the document
`c`	allow content copying or extraction
`o`	allow commenting
`f`	allow filling of form fields
`s`	allow content extraction for accessibility
`a`	allow document assembly
`d`	allow high quality printing
`-1`	(default) allow everything (all permissions are granted)
`0`	allow nothing (no permissions are granted)

The parameter 0 cannot be combined with other flags. The parameter -1 is the default, it cannot be set explicitly. In order to combine multiple permissions concatenate them to one string.

Example: The following command sets the owner password to "owner" and the permission flags to allow "printing in low resolution" and "allow form filling".

```
pdfsecure -o owner -p pf input.pdf output.pdf
```

Example: "High quality printing" requires the standard printing flag to be set too.

```
pdfsecure -o owner -p pd input.pdf output.pdf
```

-pw: Read an Encrypted PDF File

When the input PDF file is encrypted and has a user password set, (the password to open the PDF) the password can be provided as parameter of the switch `-pw`.

Example: The input PDF document is encrypted with a user password. Either the user or the owner password of the input PDF is "mypassword". The command to process such an encrypted file is

```
pdfsecure -pw mypassword input.pdf output.pdf
```

When a PDF is encrypted with a user password and the password is not provided or is incorrect, the 3-Heights™ PDF Security Shell cannot read and process the file. Instead it will generate the following error message:

```
Password wasnt correct.
```

-u: Set User Password

Set the user password of the document. If a document which has a user password is opened for any purpose (such as viewing, printing, editing), either the user or the owner password must be provided. Someone who knows the user password is able to open and read the document. Someone who knows the owner password is able to open, read and modify (e.g. change passwords) the document. A PDF document can have none, either, or both passwords.

Example: Encrypt a document with a user and an owner password.

```
pdfsecure -u userpassword -o ownerpassword input.pdf output.pdf
```

5.2 Digital Signatures

For more information on digital signatures in general, see section Digital Signatures. For more information on how to create digital signatures, see section How to Create Digital Signatures.

-ap: Signature Page Number

Set the page number of where the visual appearance of the digital signature should be placed. The default is the last page. The last page can also be set using -1 as argument.

-ar: Signature Annotation Rectangle

This option allows positioning the digital signature annotation. The default location is in the lower left corner. The units are PDF points (A4 = 595x842 points, Letter = 612x792 points).

Example: create a 200 by 60 points rectangle in the upper left corner of an A4 page

```
-cn "..." -ar 10 770 200 60
```

In order to create an invisible signature use the following rectangle:

```
-ar 0 0 0 0
```

-cn: Certificate Name (Subject)

In order to sign a PDF document, a valid, existing certificate name must be provided. Consult the chapter "Certificates" to learn more about certificates. The name of a certificate is to be provided as parameter to the -cn switch to digitally sign a PDF document as shown in the command below: This property can be used to select the signer certificate for signing (see description of Cryptographic Provider in use).

Example: Sign the document.

```
-cn "Philip Renggli"
```

The signature is added on the last page of the signed document.

-cr: Signature Reason

Add a descriptive text about the reason why the document was signed.

Example: Sign the document and add a reason text.

```
-cn "Philip Renggli" -cr "I reviewed the document"
```

The signature of the resulting output looks as shown below:

-cci: Signer contact info

Add a descriptive text as signer contact info, e.g. a phone number. This enables a recipient to contact the signer to verify the signature. This is not required in order to create a valid signature.

-ci: Certificate Issuer

The issuer of the certificate. The "Certificate Issuer" corresponds to the common name (CN) of the issuer. In the Windows' certificate store this corresponds to "Issued by". This property can be used to select the signer certificate for signing (see description of Cryptographic Provider in use).

-cno: Certificate Serial Number

Set the serial number of the certificate. Specify a hex string as displayed by the "Serial number" field in the Microsoft Management Console (MMC), e.g. "49 cf 7d d1 6c a9". This property can be used to select the signer certificate for signing (see description of Cryptographic Provider in use).

-cfp: Certificate Fingerprint

Set the hex string representation of the signer certificate's sha1 fingerprint. All characters outside the ranges 0-9, a-f and A-F are ignored. In the Microsoft Management Console, the "Thumbprint" value can be used without conversion, if the "Thumbprint algorithm" is "sha1". E.g. "b5 e4 5c 98 5a 7e 05 ff f4 c6 a3 45 13 48 0b c6 9d e4 5d f5". This property can be used to select the signer certificate for signing (see description of Cryptographic Provider in use).

-co: Do not Embed Revocation Information

This switch inhibits the embedding of revocation information such as online certificate status response (OCSP - RFC 2560) and certificate revocation lists (CRL - RFC 3280). Revocation information is either an OCSP response or a CRL, which is provided by a validation service at the time of signing and acts as proof that at the time of signing the certificate is valid. This is useful because even when the certificates expires or is revoked at a later time, the signature in the signed document remains valid.

Embedding revocation information is optional but suggested when applying advanced or qualified electronic signatures. If the embedding is enabled then the information of the signer certificate and the issuer certificates other than the root certificate is embedded as well. This implies that both OCSP responses and CRLs can be present in the same message.

The downsides of embedding revocation information are the increase of the file size (normally by around 20k) and that it requires a connection to a validation service, which delays the process of signing (normally by around 2 seconds). For mass signing it is suggested to use the caching mechanism, see chapter "Caching of CRLs, OSCP and TSP Responses".

Embedding revocation information requires an online connection to the CA that issues them. The firewall must be configured accordingly. In case a web proxy is used, it must be ensured the following MIME types are supported when using OCSP (not required for CRL):

```
application/ocsp-request
application/ocsp-response
```

-cp: Cryptographic Provider

This property specifies the cryptographic provider used to create and verify signatures.

For more information on the different providers available, see the description in the respective subsection of the section Cryptographic Provider.

- When using the Microsoft CryptoAPI Provider, the value of this property with the following syntax:
  `"[ProviderType:]Provider[;PIN]"`
  Examples, "123456" being the pin code:
  - `Provider = "Microsoft Base Cryptographic Provider v1.0;123456"`
  - `Provider = ";123456"`
- When using the PKCS#11 Provider, the value of this property is to be set to a string with the following syntax: `"PathToDll;SlotId;Pin"` Examples:
  - `Provider = "\WINDOWS\system32\siecap11.dll;4;123456"`
- When using any of the service providers, such as the Swisscom All-in signing service, the value of this property is essentially the url of the service endpoint:
  `"http[s]://server.servicedomain.com:8080/url"`

-cps: Cryptographic session property (string)

String property for configuring cryptographic session. The supported names and values are specific to the cryptographic provider.

-cpf: Cryptographic session property (file)

File data property for configuring cryptographic session. The supported names and values are specific to the cryptographic provider.

-csl: Certificate Store Location

For the Microsoft CryptoAPI Provider this defines the location of the certificate store from where the signing certificate should be taken. Supported are:

0 Local Machine

1 Current User (default)

For more information, see the detailed description of the Microsoft CryptoAPI Provider.

-csn: Certificate Store Name

For the Microsoft CryptoAPI Provider this defines the certificate store from where the signing certificate should be taken. This depends on the OS. The default is "MY". Other supported values are: "CA" or "ROOT".

Example: use the certificate store ROOT from the Local Machine account.

```
-cn "..." -csn ROOT -csl 0
```

-tsu: Time-stamp URL

The URL of the trusted Time-stamp server (TSA) from which a Time-stamp shall be acquired. This setting is only required when applying a Qualified Electronic Signature. Applying a Time-stamp requires an online connection to a time server; the firewall must be configured accordingly. In case a web proxy is used, it must be ensured the following MIME types are supported:

`application/timestamp-query`

`application/timestamp-reply`

-tsc: Time-stamp Credentials

If a Time-stamp server requires authentication, use this switch to provide the credentials.

Example: Credentials commonly have the syntax *username:password.*

```
-cn "..." -tsu http://mytimestamp.com -tsc username:password
```

-wpu: Web Proxy Server URL

In an organization where a web proxy server is in use, it must be ensured this web proxy server is specified. The URL is something like "http://proxy.example.org" or an IP address. For more information, see the chapter How to Use a Proxy.

-wpc: Web Proxy Server Credentials

If a web proxy server is used, and it requires authentication, use this switch and the syntax `user:password`.

Example: set a web proxy server URL and use authentication.

```
-wpu "http://proxy.example.org" -wpc user:password
```

For more information, see the chapter How to Use a Proxy.

-nc: Disable cache for CRL and OCSP

Get or set whether to disable the cache for CRL and OCSP responses. Using the cache is safe, since the responses are cached as long as they are valid only. The option affects both signature creation and validation.

See section on caching for more information on the caches.

-af1: Signature Font Name 1

This is the path to the font name used in upper text, i.e. the text that is set by -at1.

-af2: Signature Font Name 2

This is the path to the font name used in lower text, i.e. the text that is set by -at2.

-at1: Signature Text 1

This is the upper text that is added to the signature. If this property is set to blank, the signature name is added to the upper text line of the visual signature.

-at2: Signature Text 2

This is the lower text that is added to the signature. The text can be multi-lined by using carriage returns. If this property is set to blank, a three-line text is constructed that consists of:

- A statement who applied to signature
- The reason of the signature
- The date

-abg: Signature Background Image

This is the background image that is added to the signature. The image is centered and scaled down proportionally to fit into the given rectangle. If the path is NULL, or the image does not exist, the appearance's background is a filled rectangle using the colors fill color and stroke color. Note that for the output file to be PDF/A, the image's color space must match the document's output intent.

In order to create a signature with the image only, set the signature texts 1 and 2 to " ".

-al: Signature Line Width

This is the thickness of the line surrounding the visual signature in points.

-acf: Signature Fill Color

This is the color of the signature's background as in RGB value. The default is 16761024 (red = 192, green = 192, blue = 255). In order to not set a color, i.e. keep the rectangle transparent, set it to -1. Color examples (color = red + green * 256 + blue * 256*256, where red, green and blue assume values from 0 to 255):

Red = 255,0,0
Cyan = 0,255,255
Black = 0,0,0
Green = 0,255,0
Magenta = 255,0,255
White = 255,255,255
Grey = 128,128,128
Blue = 0,0,255
Yellow = 255,255,0

-acs: Signature Stroke Color

This is the color of the signature's border line as RGB value. The default is 8405056 (red = 64, green = 64, blue = 128). In order to not set a color, i.e. keep it transparent, set it to -1.

-dts: Create a Time-stamp Signature

Add a document level Time-stamp. No appearance is created. The following signature options must be set: -tsu. The following signature options may be set: -cp, -tsc, -wpu, -wpc.

-mdp: Create a DocMDP Signature

This option creates a DocMDP (document modification detection and prevention) signature instead of a document signature. The DocMDP signature is also referred to as "certify a document". Note: This version can create visible DocMDP signatures. In order to create an invisible signature, set the signature's rectangle as follows: -ar 0 0 0 0.

-dap: Document Access Permissions for DocMDP Signature

This option controls the type of permitted modifications to a certified document. Valid values are:

1. No changes to the document are permitted; any change to the document invalidates the signature (default).
2. Permitted changes are filling in forms, instantiating page templates, and signing; other changes invalidate the signature.
3. Permitted changes are the same as for 2, as well as annotation creation, deletion, and modification; other changes invalidate the signature.

-fs: Force Signature

Force signature allows DocMDP (PDF 1.6) and Time-stamp signatures (PDF 2.0) on PDF/A-1 documents. The output file's version is upgraded and PDF/A conformance removed. So the output file will contain the signature, but not be PDF/A-1 anymore. Applying a DocMDP or Time-stamp signature breaks PDF/A-1 compliance, therefore the default behavior is to abort the operation with an error.

-p2f: Replace placeholder image with signature field

This option enables the replacement of special placeholder images with signature fields. This function is used to automatically place signature fields under the control of the creator program.

-vs: Verify signature

This option verifies all signatures in the input document. Get more information on the signatures by using the option -v.

For more information on validating digital signature, see the section sec:howtovalidatesig]How to Validate Digital Signatures.

5.3 General Switches

-lk: Set License Key

Pass a license key to the application at runtime instead of installing it on the system.

-s: Add Stamps

Add a stamp xml file. For more information about stamping, see the chapter Stamping.

-v: Verbose Mode

This option turns on the verbose mode.

In the verbose mode, the steps performed by 3-Heights™ PDF Security Shell are written to standard output

Example: Enable the verbose mode.

```
pdfsecure -v in.pdf  out.pdf
processing file in.pdf.
Done
```

5.4 Return Codes

All return codes other than 0 indicate an error in the processing.

Table: Return Codes	
Value	**Description**
0	Success
1	Couldn't open input file
2	PDF Output File could not be created
3	Error with switch or too many parameters
4	PDF Input file is encrypted and password is missing or incorrect
5	Cannot create signature
6	Cannot get response from OCSP or TSP
7	Input file contains invalid signatures (validation)
10	License Error

Possible reasons for return code 5 are:

- Cannot create a session (or CSP)
- The certificate store is not available
- The certificate cannot be found
- The private key is not available
- Incorrect signature length

Return code 6 can be considered as a warning, i.e. non-critical. It means the document was correctly signed, but due to a HTTP missing connection to the OCSP or time server it does not contain either revocation information or a Time-stamp.

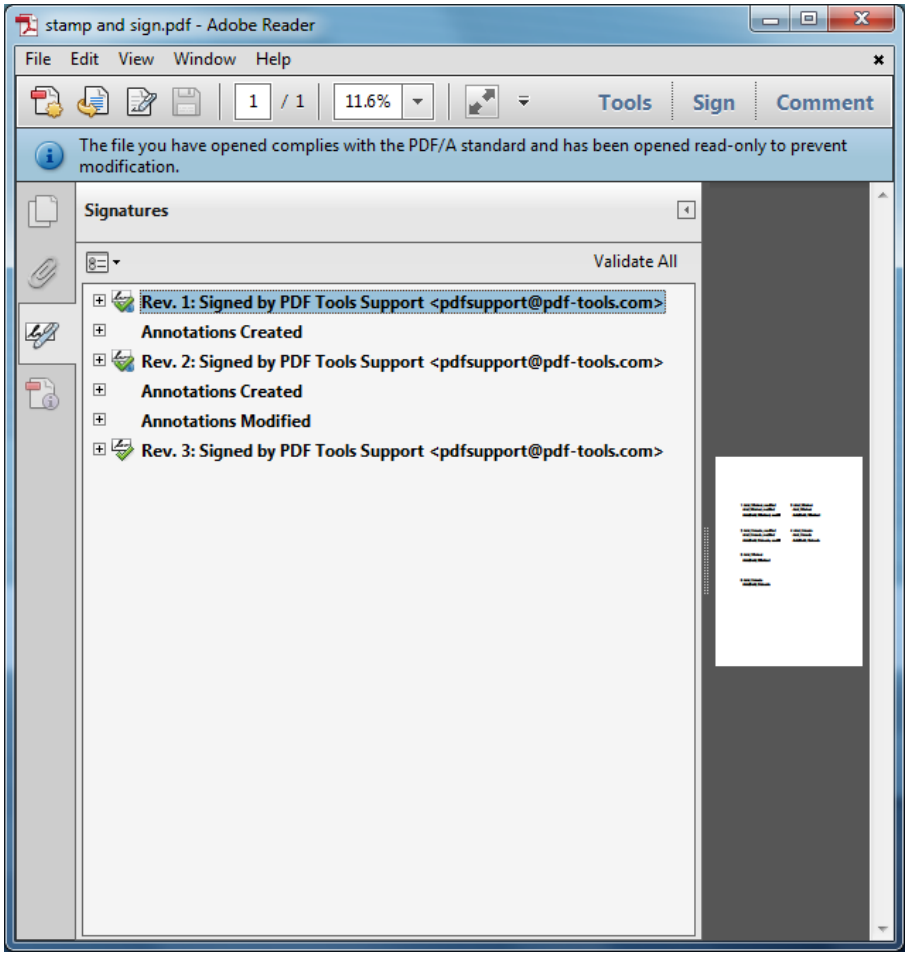
Frequent Error Source

It may happen that you type a command, or copy it from this manual and it doesn't work even though it seems to be correct. A common reason is that the dash (-) which is used for most parameters is accidently mistaken by an em dash (—).

6 Stamping

The 3-Heights™ PDF Security Shell can add new content such as text or images to the output document. This process is called stamping. The content of previously applied stamps can be modified.

The 3-Heights™ PDF Security Shell can sign and stamp documents in one step. In order to not invalidate existing signatures, stamps can be modified and created using stamp annotations with an incremental update to the input document. An example of this can be seen in the screenshot below.



Stamp File Syntax

Stamps are described with xml data that is passed to the 3-Heights™ PDF Security Shell as file using the option `-s`.

A stamp file can contain one or more stamps. Each stamp is defined by a `ps:stamp` tag that specifies the stamp's size, position, and pages to which it is applied to.

Each stamp contains a number of content operators that define the appearance (i.e. the content) of the stamp. The content operators are applied in the order they appear within `ps:stamp` where each content element is drawn over all previous elements (i.e. increasing z-order).

Tags, Attributes-Names, Attribute-Values

`ps:pdfstamp`	The *Root Tag* for the PDF stamps. The tag may contain multiple stamps.
`xmln:ps http://www.pdf-tools.com/pdfstamp/`	
`ps:stamp`	*Stamp*: A stamp contains a collection of content operators.
`page first, even, odd`	The pages to which the stamp is to be applied. Comma-separated combinations are allowed. first First page last Last page odd Only odd pages including first page and last page in case it is odd even Only even pages including last page in case it is even all All pages not_first First page excluded not_last Last page excluded
`name name`	Identifier of the stamp optional, must be less than 127 characters, must be unique, see note 2 below.
`relativepos x y`	Relative position x and y of the stamp with regards to the page. Positive values of x and y define the distances of the stamp to the left and lower, negative values to the right and upper page boundary respectively. The positioning algorithm works best for stamp rotation angles that are a multiple of 90 (see rotate attribute). x or y is ignored, if respective align is used. Example 1: `relativepos=" 10 -10"` places stamp in upper left corner of page Example 2: `relativepos="-10 -10"` places stamp in upper right corner of page Example 3: `relativepos=" 10 10"` places stamp in lower left corner of page Example 4: `relativepos="-10 10"` places stamp in lower right corner of page
`align center middle`	Align the stamp with the page center position horizontally at center of page, the x value of relativepos is ignored and should be set to 0. middle position vertically at middle of page, the y value of relativepos is ignored. Example 1: `position="0 4" align="center"` centers the stamp horizontally and 4pt away from the bottom of the page. Example 2: `position="-4 0" align="middle"` centers the stamp vertically and 4pt away from the right edge of the page.
`size w h`	The width and height of the stamp. The stamp's content will be clipped to this rectangle. If this is not specified or either w or h are zero, the respective size is calculated to fit content.

`rotate r`	Rotation of the stamp in degrees clockwise.
`scale relToA4`	Scale the stamp relative to the page size. i.e. make stamp half as large on a A5 and twice as large on a A3 page as specified.
`autoorientation true`	Detect orientation (portrait and landscape) of page automatically and treat landscape page as 90° rotated portrait. Useful to apply stamps to "long" or "short" edge of page.
`alpha t`	The opacity of the stamp as a whole. 1.0 for fully opaque, 0.0 for fully transparent. Default: 1.0 The PDF/A-1 standard does not allow transparency. Therefore, for PDF/A-1 conforming input files you must not set alpha to a value other than 1.0.
`type type`	The type of the stamp `annotation` (default): The stamp is added to the page as a stamp annotation. Creating or modifying stamps of this type will not invalidate existing signatures of the input document. While it is not easily possible to remove stamps of this type, it is possible to print a document without annotations. `foreground`[7]: The stamp is added to the foreground of the page content. Creating or modifying stamps of this type will invalidate all existing signatures of the input document. It is not easily possible to remove stamps of this type nor can the document be printed without them. `background`: The stamp is added to the background of the page content. Creating or modifying stamps of this type will invalidate all existing signatures of the input document. It is not easily possible to remove stamps of this type nor can the document be printed without them. Note that stamps placed this way can be hidden when pages contain a non-transparent background. In these cases, you may rather want to put the stamps in the foreground, but apply alpha transparency to achieve a result with existing content not covered completely.
`ps:rotate`	*Rotate*: Applies to stamp content defined within this tag
`angle n`	Rotate by n degrees counter-clockwise, e.g: 90
`origin x y`	Set the origin of the rotation in points, e.g. 100 100
`ps:translate`	*Coordinate Translation*: Applies to stamp content defined within this tag
`offset x y`	The x (horizontal) and y (vertical) offset in points. A translation by x y is equal to a transformation by 1 0 0 1 x y.
`ps:transform`	*Coordinate Transformation*: Applies to stamp content defined within this tag
`matrix a b c d x y`	The transformation matrix, to scale rotate, skew, translate, etc the stamp, e.g: Identity: 1 0 0 1 0 0 Scale by factor 2: 2 0 0 2 0 0 Translate 50 points left, 200 up: 1 0 0 1 50 200 Rotate by x: cos(x) sin(x) -sin(x) cos(x) 0 0 For 90 ($= \pi/2$) that is: 0 1 -1 0 0 0

[7] Up to version 4.5.6.0 of the 3-Heights™ PDF Security Shell this type was called content.

ps:filltext	*Add Filled Text*
color r g b	The color as RGB value, where all values must be from 0 to 1, e.g: Red: 1 0 0 Green: 0 1 0 Black: 0 0 0 Gray: 0.5 0.5 0.5
position x y	The position in points in the stamp, e.g. 200 300 With the default align values (`align="left top"`), position defines the left top corner of the text.[8]
align xalign yalign	Align text at position or stamp, if position is not set. Values for horizontal alignment xalign: - left align to the left (default) - center center text - right align to the right Values for vertical alignment yalign: - top align to the top (default) - middle align to the middle - bottom align to the bottom Example: `align="left bottom"` positions the text in the left bottom corner of the stamp. If position is set, align left bottom corner of text to position.
font name	The true type name of the font, e.g. `Arial` or `Times New Roman,Bold`, or a complete path to the font, e.g. `C:/Windows/Fonts/Arial.ttf`. If the name is used, the respective font must be available in any of the font directories (see chapter Fonts).
size n	The font size in points, e.g. 12. If set to 0 , the size is chosen such that text fits stamp size (not allowed, if operator is within transformation operator).
fontencoding encoding	Encoding of the font. Allowed values are Unicode (default) and WinAnsi, see note 3 below
text text	The text that is to be written, e.g. `text="Hello World"` Multiline text is supported by using the newline character `
`, e.g. `text="1st line
2nd line"`.
ps:stroketext	*Add Stroked Text* (Outlined Text): For parameters see filltext.
linewidth f	Set the linewidth in points, e.g. 1.0.
color r g b	See filltext
position x y	See filltext
align xalign yalign	See filltext
font name	See filltext
size n	See filltext
fontencoding encoding	See filltext
text text	See filltext

[8] Prior to version 4.4.31.0 of the 3-Heights™ PDF Security Shell, position specified the origin of the first character. When upgrading, add `0.75*size` to the value of y.

`ps:image`	*Add Image*: In order for the stamp to be PDF/A compliant, the image's color space must match the document's output intent's.
`rect x y w h`	The rectangle where the image is to be placed at. x,y correspond the the location (origin at lower left corner), and w,h to width and height, e.g: 100 200 50 50
`filename path`	The path to the file, e.g: C:\pictures\image1.jpg
`compression value`	By default bi-tonal images are compressed with CCITTFax, continuous tone images with DCT and indexed images with Flate. To explicitly set the compression use this property. Support values are: `Flate` Flate encoded `DCT` DCT (JPEG) encoded `CCITTFax` CCITT G4 encoded
`ps:fillrectangle`	*Add Filled Rectangle*:
`rect x y w h`	The coordinates and size of the rectangle. If this value is omitted, the rectangle fills the area of the stamp.
`color r g b`	The fill color of the rectangle. The color as RGB value, where all values must be from 0 to 1.
`alpha t`	The opacity of the rectangle. 1.0 for fully opaque, 0.0 for fully transparent. Default: 1.0 The PDF/A-1 standard does not allow transparency. Therefore, for PDF/A-1 conforming input files you must not set alpha to a value other than 1.0.
`ps:strokerectangle`	*Add Stroked Rectangle*:
`rect x y w h`	See fillrectangle
`linewidth f`	Set the linewidth in points, e.g. 1.0.
`color r g b`	The stroke color of the rectangle. The color as RGB value, where all values must be from 0 to 1.
`alpha t`	See fillrectangle

Notes

1. All coordinate and size values are in PDF units of 1/72 inch (A4 = 595 x 842 points, letter = 612 x 792 points). The origin of the coordinate system is generally the lower left corner of the reference object. For stamps the reference object is the page, for content operators the reference is the stamp rectangle.
2. Setting the name attribute of a stamp allows the stamp's content to be replaced later. If an existing stamp with the same name exists in the input file, its content is replaced. Otherwise a new stamp is created. Note that when updating a stamp, its position and size remains. Therefore, if you intend to update a stamp, make sure to create it using a size that is sufficiently large.
3. Attribute fontencoding: The PDF/A standard demands that all used fonts must be embedded in the PDF. Since fonts with many glyphs can be very large in size (>20MB), unused glyphs are removed prior to embedding. This process is called subsetting. The attribute fontencoding controls the subsetting:
   - Unicode: Only the glyphs used by the stamp are embedded. If the stamp is modified, a new font that includes the new glyph set has to be re-embedded. This setting is recommended for stamps that need not be modified later.
   - WinAnsi: All glyphs required for WinAnsiEncoding are embedded. Hence the text's characters must be limited to this character set. If the content of the stamp is updated, fonts using WinAnsi will be reused.

   For example, embedding the font "Arial" with Unicode and approximately ten glyphs uses 20KB while "Arial" with WinAnsi (approximately 200 glyphs) uses 53KB of font data.

Example: Simple Stamps

Apply two simple stamps.

First Stamp: Stamp text “Simple Stamp” on in upper right corner of all pages.

Second Stamp: Stamp image image.jpg rotated by 90° at the corner of the top edge of the first page.

```
<?xml version="1.0" encoding="utf-8"?>
<ps:pdfstamp xmlns:ps="http://www.pdf-tools.com/pdfstamp/">
  <ps:stamp page="all" name="simple stamp"
   relativepos="10 -10" size="160 0">
     <ps:filltext align="left middle" fontencoding="WinAnsi"
      font="Arial" size="12"
      text="Simple Stamp" />
  </ps:stamp>
  <ps:stamp page="first" relativepos="0 -10" align="center">
    <ps:rotate angle="90" origin="50 50">
      <ps:image rect="0 0 100 100"
       filename="c:\images\image.jpg"/>
    </ps:rotate>
  </ps:stamp>
</ps:pdfstamp>
```

Example: Modify Simple Stamp

Modify simple stamp from example above.

The stamp `simple stamp` can be modified by applying the following stamp xml file to the output file of the example above. Note that since position and size of the stamp remain unchanged, the respective attributes can be omitted.

```
<?xml version="1.0" encoding="utf-8"?>
<ps:pdfstamp xmlns:ps="http://www.pdf-tools.com/pdfstamp/">
  <ps:stamp name="simple stamp">
    <ps:filltext align="left middle" fontencoding="WinAnsi"
     font="Arial" size="12"
     text="Modified Stamp" />
  </ps:stamp>
</ps:pdfstamp>
```

Example: Add a watermark text diagonally across page

Note: an A4 page is 595 by 842 points.

```
<?xml version="1.0" encoding="utf-8"?>
<ps:pdfstamp xmlns:ps="http://www.pdf-tools.com/pdfstamp/">
  <ps:stamp page="all" size="595 842"
   align="center middle"
   scale="relToA4" autoorientation="true"
   type="foreground">
     <ps:rotate angle="55" origin="298 421">
       <ps:stroketext align="center middle" position="298 421"
        font="Arial,Bold" size="60"
        text="WATERMARK TEXT"/>
     </ps:rotate>
  </ps:stamp>
</ps:pdfstamp>
```

Example: Apply stamp to long edge of all pages

Stamp has a light gray background and a black border.

```
<?xml version="1.0" encoding="utf-8"?>
<ps:pdfstamp xmlns:ps="http://www.pdf-tools.com/pdfstamp/">
  <ps:stamp page="all" size="802 12"
   relativepos="5 0" align="middle" rotate="90"
   scale="relToA4" autoorientation="true"
   alpha="0.75" type="foreground">
     <ps:fillrectangle color="0.8 0.8 0.8" />
     <ps:strokerectangle />
     <ps:filltext align="center middle"
      font="Arial" size="8"
      text="stamp on long endge"/>
  </ps:stamp>
</ps:pdfstamp>
```

7 Licensing and Copyright

The 3-Heights™ PDF Security Shell is copyrighted. This user's manual is also copyright protected; it may be copied and given away provided that it remains unchanged including the copyright notice.

8 Contact

PDF Tools AG

Kasernenstrasse 1

8184 Bachenbülach

Switzerland

http://www.pdf-tools.com